T-79.159 Cryptography and Data Security

Introduction to Cryptography

Helger Lipmaa
Laboratory for Theoretical Computer Science
Helsinki University of Technology
helger@tcs.hut.fi
http://www.tcs.hut.fi/~helger

Cryptography and Data Security / 2004

- Lecturer: Helger Lipmaa
- Reception: by appointment
- Lectures and recommended exercise sessions
- Course material: Slides
- Newsgroup: opinnot.tik.salaus

Comparison with T-79.159/2003

- Slides from 2003 are on the web
- Can use for "early learning", except that:
- Slides will be corrected (bugs + made more readable)
- There will be at least one extra lecture
- Reference book for 2003, *Network Security* (Kaufman, Perlman, Speciner), is still usable but not required

Goals

- Introduction to cryptography and its methods
- To give basic overview of existing primitives and protocols
- To explain which tasks and how can be performed securely and which tasks can be not
- To understand what it means for something to be secure
- Hopefully: To develop basic cryptographic thinking

What this course is (not) about?

- *Not* about politics, coorporate security
- *Not* about database security, intrusion detection — university has other courses for that
- *Not much* about applications like PGP
- *Is* about cryptography, the mathematical part of cryptography
- *Is* about novel uses of cryptography (e-voting, . . .)

Prerequisities

- Mathematics: one or two years of basic studies + Mat-1.128 (or an analogue). Discrete mathematics is essential!
- Understanding of computer architectures
- Coding skills: some home assignments will need programming
- Some basic knowledge about data security
- Sophisticated and curious mind. Interest in solving puzzles, security issues

Course Team

- Lectures: Helger Lipmaa (English + some other obscure languages)
- Tutorials: Markku-Juhani Saarinen (Finnish + English + . . .)

Course Layout

- More or less follow the textbook during approx. the first seven lectures
- New and interesting stuff in last lectures
- Students can buy the textbook (has been spotted in Akateeminen), but it is not necessary

Tentative Schedule

$\sharp$	Date	Subject
1.	21.1	Introduction (Chapter 2)
2.	28.1	Secret key Cryptography (Chp 3)
3.	4.2	Hash functions (Chp 5) — MJOS
4.	11.2	Block cipher modes (Chp 4)
5.	18.2	Public key algorithms (Chp 6)
6.	25.2	Identification (roughly Chp 7)
7.	3.3	. . . [new] — MJOS
8.	10.3	Zero-knowledge and commitments
9.	17.3	Secret sharing, threshold encryption, MPC
10.	7.4	Pseudorandomness, provable security
11.	14.4	Electronic cash
11.	21.4	. . . [new]
12.	28.4	Epilogue

Course Passing

- 12 lectures, 11 tutorials — when lecture is on Wednesday, the corresponding tutorial (homework) will be available on Monday and the exercise session will be held on Thursday (of the next week)
- Thus, first exercise session: 29.01
- Homeworks checked by MJOS (B254, mjos at tcs.hut.fi) during the exercise session
- To get to exam, 50% of the homeworks must be passed (6 of 11)
- Exam — time not fixed yet

First Lecture: Introduction to Cryptography

1. What is cryptography?
2. Breaking an encryption scheme
3. Types of cryptographic functions
4. Secret key cryptography
5. Public key cryptography
6. Hash algorithms

(Chapter 2)

What is cryptography?

- $\kappa\rho\upsilon\pi\tau o$-$\gamma\rho\alpha\phi\eta$ = hidden + writing
- Historically, cryptography = the science of secret communication (encryption)
- Alice and Bob want to communicate without the governmental interception
- Two governments want to communicate without any interception whatsoever

What is cryptography?

- Apart from encryption, contemporary cryptography makes it possible to
  - ⋆ authenticate people,
  - ⋆ verify the integrity of data
  - ⋆ . . . (many unexpected applications)
- Communication of *digital* information (encoded as numbers)
- Different functions map numbers other numbers either to encrypt them, to authenticate, . . .

Need for the Key

- Ciphertext = encrypted plaintext (message), $C = E(M)$
- Plaintext = decrypted ciphertext, $M = E^{-1}(C)$
- Function E^{-1} must be secret—otherwise it is easy to compute M from C
- If Alice and Bob want to have twodirectional traffic, they must share the function E (and E^{-1}) — a hardware module, piece of software or a mathematical description

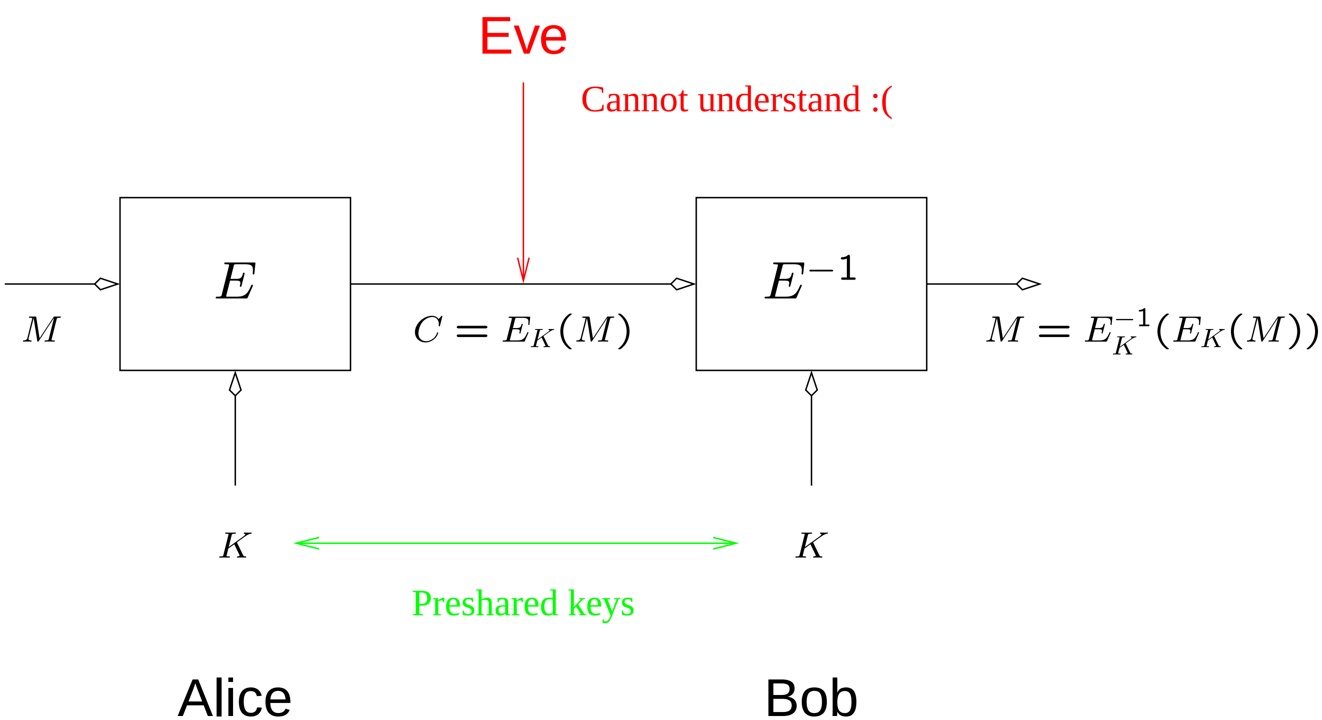
The Need for the Key

- Bad 1: the description of E might be long, and hard to share
- Bad 2: the description of E might be long, and hard to keep in secret
- E.g., can be recovered by reverse engineering the hardware module
- Solution: E and E^{-1} are public, but C also depends on a *short* secret key K
- Easier to share, easier to keep secret (memorize, or store in tamper-proof hardware)

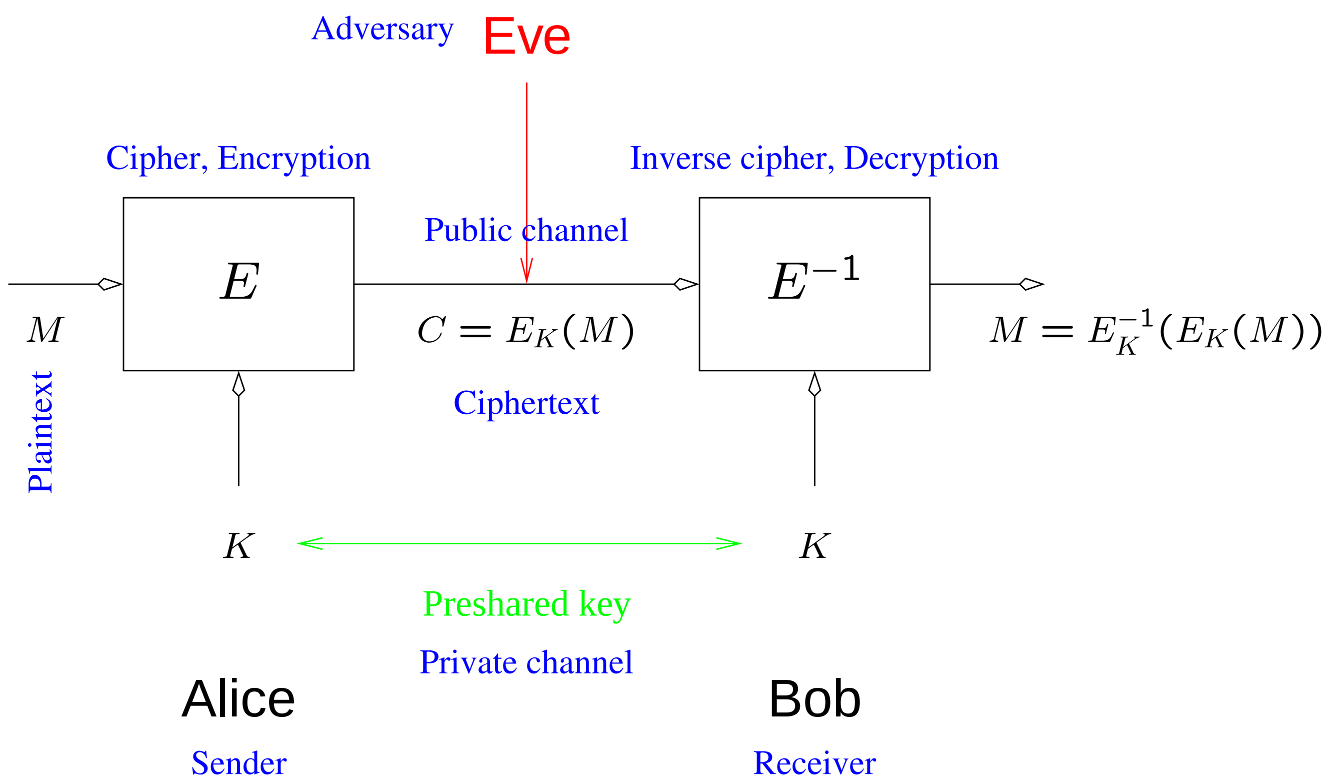
Types of cryptographic functions

- Secret key cryptography: 1 key
- Public key cryptography: 2 keys
- Hash functions: no keys

Secret key encryption: basic model



Encryption: definitions



Scientific method of cryptography

- Security of cryptographic primitives is either
  - ⋆ Provable: e.g., one-time pad is secure
  - ⋆ Reducable: "E is secure if F is secure"
  - ⋆ Heuristic: "we cannot break E, and a lot of other people also do not know how to break it"

- Fundamentally, it is not known if *any* cryptographic method is secure — since it might happen that $\mathbf{P} = \mathbf{NP}$, or that quantum computers can break all ciphers

Scientific method of cryptography

- *Provable*: most desired, but such systems cannot be practical
- *Reducable*: practical in some applications, but usually slow and one must have secure basic primitives
- *Heuristic*: results in crazy but extremely practical ciphers
- It is also not easy to define what exactly is meant by security in practice!
- The real method: Alice designs a cipher, Bob breaks it, Alice fixes the break, Carol breaks it, Alice and Diana fix the break, Edward breaks it, . . . , Theodor proposes a completely new cipher, Urho breaks it, . . .

Ciphers should be public, 1/2

- If cipher is kept secret, it may be harder to break it
- However, one cannot rely on secrecy: the more people use a cipher, the more information about it is bound to leak
- Main reason for publishing: gives free scientific scrutinity
- Avoids also criticism

Ciphers should be public, 2/2

- People will try to break your cipher (for their personal fame, for hobby, for . . .). If they cannot break it in a while, the cipher might be secure

- If you know the cipher is secure anyways (i.e., not heuristic), then publishing it does not help to break it!

- Motivations for keeping it secret: (a) trade secrets, (b) NSA/KGB/. . . develops a secure cipher and does not want others to start use it

Computational difficulty

- Encrypting and decrypting, if you know the key, must be easy
- That is, functions E and E^{-1} are efficient
- In practice, E's time complexity is required to be linear/quadratic in the length of key
- Recovering the key you don't know must be difficult
- Exhaustive key search: If key length is k bits, there are 2^k keys
- Therefore e.s. takes 2^k steps

Computational difficulty: by example

- Locker has k decimal digits. Setting one digit takes 1 second if you know it

- Total effort for "decrypting": k seconds

- Bad guy must try up to 10^k combinations, thus 10^k seconds

- Increasing k by one increases your effort by one second, and the effort of the bad guy 10 times

- Increase k from 10 to 11: you spend one more second, bad guy spends 70000 more years

Computational difficulty: by example

A catch:

- Of course, the attacker can opt to use a bolt cutter. . .

The famous Caesar cipher

- Plaintext consists of the letters A, . . . , Z
- When computing a ciphertext, “add 3” (modulo 26) to all letters
- That is, A $\rightarrow$ D, B $\rightarrow$ E, . . . , X $\rightarrow$ A, Y$\rightarrow$B, Z$\rightarrow$ A
- Do it for every letter
- Example: CAESAR $\rightarrow$ FDHVDU
- Security depends on the cipher to be secret. Once you know the cipher, you can decrypt everything

Shift ciphers

- Pick a secret key K from 0 to 25. Add K modulo 26 to all letters: $C = M + K \mod 26$
- Example: if $K = 1$ then $E_K(\text{IBM}) = \text{HAL}$
- Increased security: even if cipher becomes public, there is still 26 keys

Shift ciphers: Cryptanalysis

- Statistical cryptanalysis, based on frequency of letters. If the original message is redundant (e.g., written in English), then also the ciphertext will be redundant

- In long plaintexts, the frequency of different letters is close to the well-known frequency of different letters in average English texts. Since there is a one-to-one mapping between plaintext and ciphertext letters, recovering the plaintext is easy

Frequency table of English Letters

Letter	Freq	Letter	Freq.
A	0.082	N	0.067
B	0.015	O	0.075
C	0.028	P	0.019
D	0.043	Q	0.001
E	0.127	R	0.060
F	0.022	S	0.063
G	0.020	T	0.091
H	0.061	U	0.028
I	0.070	V	0.010
J	0.002	W	0.023
K	0.008	X	0.001
L	0.040	Y	0.020
M	0.024	Z	0.001

Example

Ciphertext 1: `gth`

Ciphertext 2: `hxdjannrcqnafrcqdbxajpjrwbcdbrwcqnorpqcjpjrwbccnaaxa`

Write down all 26 possible decryptions, see if you can spot one that makes sense!

Example

Ciphertext:	hxdjannrcqnafrcqdbxajpjrwbcdbrwcqnorpqcjpjrwbccnaaxa
k= 0	hxdjannrcqnafrcqdbxajpjrwbcdbrwcqnorpqcjpjrwbccnaaxa
k= 1	gwcizmmqbpmzeqbpcawzioiqvabcaqvbpmnqopbioiqvabbmzzwz
k= 2	fvbhyllpaolydpaobzvyhnhpuzabzpuaolmpnoahnhpuzaalyyvy
k= 3	euagxkkoznkxcoznayuxgmgotyzayotznklomnzgmgotyzzkxxux
k= 4	dtzfwjjnymjwbnymzxtwflfnsxyzxnsymjknlmyflfnsxyyjwwtw
k= 5	csyeviimxlivamxlywsvekemrwxywmrxlijmklxekemrwxxivvsv
k= 6	brxduhhlwkhuzlwkxvrudjdlqvwxvlqwkhiljkwdjdlqvwwhuuru
k= 7	aqwctggkvjgtykvjwuqtcickpuvwukpvjghkijvcickpuvvgttqt
k= 8	zpvbsffjuifsxjuivtpsbhbjotuvtjouifgjhiubhbjotuufssps
k= 9	youareeitherwithusoragainstusinthefightagainstterror
k=10	xntzqddhsgdqvhsgtrnqzfzhmrstrhmsgdehfgszfzhmrssdqqnq
k=11	wmsypccgrfcpugrfsqmpyeyglqrsqglrfcdgefryeyglqrrcppmp
k=12	vlrxobbfqebotfqerploxdxfkpqrpfkqebcfdeqxdxfkpqqboolo
k=13	ukqwnaaepdansepdqoknwcwejopqoejpdabecdpwcwejoppannkn
k=14	tjpvmzzdoczmrdocpnjmvbvdinopndioczadbcovbvdinoozmmjm
k=15	sioulyycnbylqcnbomiluauchmnomchnbyzcabnuauchmnnyllil
k=16	rhntkxxbmaxkpbmanlhktztbglmnlbgmaxybzamtztbglmmxkkhk
k=17	qgmsjwwalzwjoalzmkgjsysafklmkaflzwxayzlsysafkllwjjgj
k=18	pflrivvzkyvinzkyljfirxrzejkljzekyvwzxykrxrzejkkviifi
k=19	oekqhuuyjxuhmyjxkiehqwqydijkiydjxuvywxjqwqydijjuhheh
k=20	ndjpgttxiwtglxiwjhdgpvpxchijhxciwtuxvwipvpxchiitggdg
k=21	mciofsswhvsfkwhvigcfouowbghigwbhvstwuvhouowbghhsffcf
k=22	lbhnerrvgurejvguhfbentnvafghfvagursvtugntnvafggreebe
k=23	kagmdqquftqdiuftgeadmsmuzefgeuzftqrustfmsmuzeffqddad
k=24	jzflcpptespchtesfdzclrltydefdtyespqtrselrltydeepcczc
k=25	iyekboosdrobgsdrecybkqksxcdecsxdropsqrdkqksxcddobbyb

Example

Ciphertext:	hxdjannrcqnafrcqdbxajpjrwbcdbrwcqnorpqcjpjrwbccnaaxa
k= 0	hxdjannrcqnafrcqdbxajpjrwbcdbrwcqnorpqcjpjrwbccnaaxa
k= 1	gwcizmmqbpmzeqbpcawzioiqvabcaqvbpmnqopbioiqvabbmzzwz
k= 2	fvbhyllpaolydpaobzvyhnhpuzabzpuaolmpnoahnhpuzaalyyvy
k= 3	euagxkkoznkxcoznayuxgmgotyzayotznklomnzgmgotyzzkxxux
k= 4	dtzfwjjnymjwbnymzxtwflfnsxyzxnsymjknlmyflfnsxyyjwwtw
k= 5	csyeviimxlivamxlywsvekemrwxywmrxlijmklxekemrwxxivvsv
k= 6	brxduhhlwkhuzlwkxvrudjdlqvwxvlqwkhiljkwdjdlqvwwhuuru
k= 7	aqwctggkvjgtykvjwuqtcickpuvwukpvjghkijvcickpuvvgttqt
k= 8	zpvbsffjuifsxjuivtpsbhbjotuvtjouifgjhiubhbjotuufssps
k= 9	**youareeitherwithusoragainstusinthefightagainstterror**
k=10	xntzqddhsgdqvhsgtrnqzfzhmrstrhmsgdehfgszfzhmrssdqqnq
k=11	wmsypccgrfcpugrfsqmpyeyglqrsqglrfcdgefryeyglqrrcppmp
k=12	vlrxobbfqebotfqerploxdxfkpqrpfkqebcfdeqxdxfkpqqboolo
k=13	ukqwnaaepdansepdqoknwcwejopqoejpdabecdpwcwejoppannkn
k=14	tjpvmzzdoczmrdocpnjmvbvdinopndioczadbcovbvdinoozmmjm
k=15	sioulyycnbylqcnbomiluauchmnomchnbyzcabnuauchmnnyllil
k=16	rhntkxxbmaxkpbmanlhktztbglmnlbgmaxybzamtztbglmmxkkhk
k=17	qgmsjwwalzwjoalzmkgjsysafklmkaflzwxayzlsysafkllwjjgj
k=18	pflrivvzkyvinzkyljfirxrzejkljzekyvwzxykrxrzejkkviifi
k=19	oekqhuuyjxuhmyjxkiehqwqydijkiydjxuvywxjqwqydijjuhheh
k=20	ndjpgttxiwtglxiwjhdgpvpxchijhxciwtuxvwipvpxchiitggdg
k=21	mciofsswhvsfkwhvigcfouowbghigwbhvstwuvhouowbghhsffcf
k=22	lbhnerrvgurejvguhfbentnvafghfvagursvtugntnvafggreebe
k=23	kagmdqquftqdiuftgeadmsmuzefgeuzftqrustfmsmuzeffqddad
k=24	jzflcpptespchtesfdzclrltydefdtyespqtrselrltydeepcczc
k=25	iyekboosdrobgsdrecybkqksxcdecsxdropsqrdkqksxcddobbyb

Substitution ciphers

- Key K is an arbitrary permutation of the set A, . . . , Z
- Since there are $26! = 26 \cdot 25 \cdot 24 \cdots 1 \approx 2^{88}$ such keys, writing down all decryptions is impossible
- Statistical methods still apply

Breaking an encryption scheme

- Ciphertext-only attacks
- Known plaintext attacks
- Chosen plaintext attacks
- Fancy stuff

Ciphertext-only attack

- Given sufficiently long ciphertext, so that you can perform statistical analysis

- Needed: long ciphertext

- Needed: extremely weak cipher (like a substitution cipher)

Known-plaintext attack

- Often the attacker gets to know the plaintexts that correspond to some ciphertexts
- Many reasons: encrypted IP packets have known header, encrypted emails start with a “Dear“, . . .
- This should not help in finding the key
- Substitution ciphers extremely weak: if you know the encryptions of some of the most frequent letters, you can often guess the rest
- Stronger than a ciphertext-only attack

Chosen-plaintext attack

- In many applications, the attacker is able to encrypt a few chosen plaintexts. She should not be able to decrypt your (different) messages later
- Example: Eve gets your smartcard for a five minutes, and encrypts some random messages. In substitution cipher, encrypt the message "The quick brown fox jumps over the lazy dog"
- Stronger than a known-plaintext attack
- Good cipher is employed everywhere: thus should be secure at least against a chosen-plaintext attack

Beyond CPA

- Implementation attacks: faulty implementations, timing attacks, power attack

- Related key attacks

- Distinguishing attacks

Secret key cryptogaphy: Uses

- Transmitting over an insecure channel
- Secure storage on insecure media
- Authentication
- Integrity check

Secret key identification

- Alice and Bob share a secret key, and want to identify each other
- Idea: “show” that you know the key but without “revealing” it
- Simple idea: Alice sends a random challenge to Bob, who sends its encryption back to Alice. Alice is thus convinced that Bob knows the secret key. Switch the roles
- Actual protocols are more complicated

Network Security calls this “authentication”. Identification is the correct term

Message authentication

- Alice and Bob share a secret key. After getting a message M from network, Bob wants to be sure it comes from Alice

- Alice authenticates the message by applying a secret key MAC $\mathbf{MAC}$ to M: $Tag = \mathbf{MAC}_K(M)$

- Bob applies a special verification algorithm to Tag to check whether $Tag =^? \mathbf{MAC}_K(M)$

- Some MACs are based on ciphers, some are not

Public-Key Cryptography

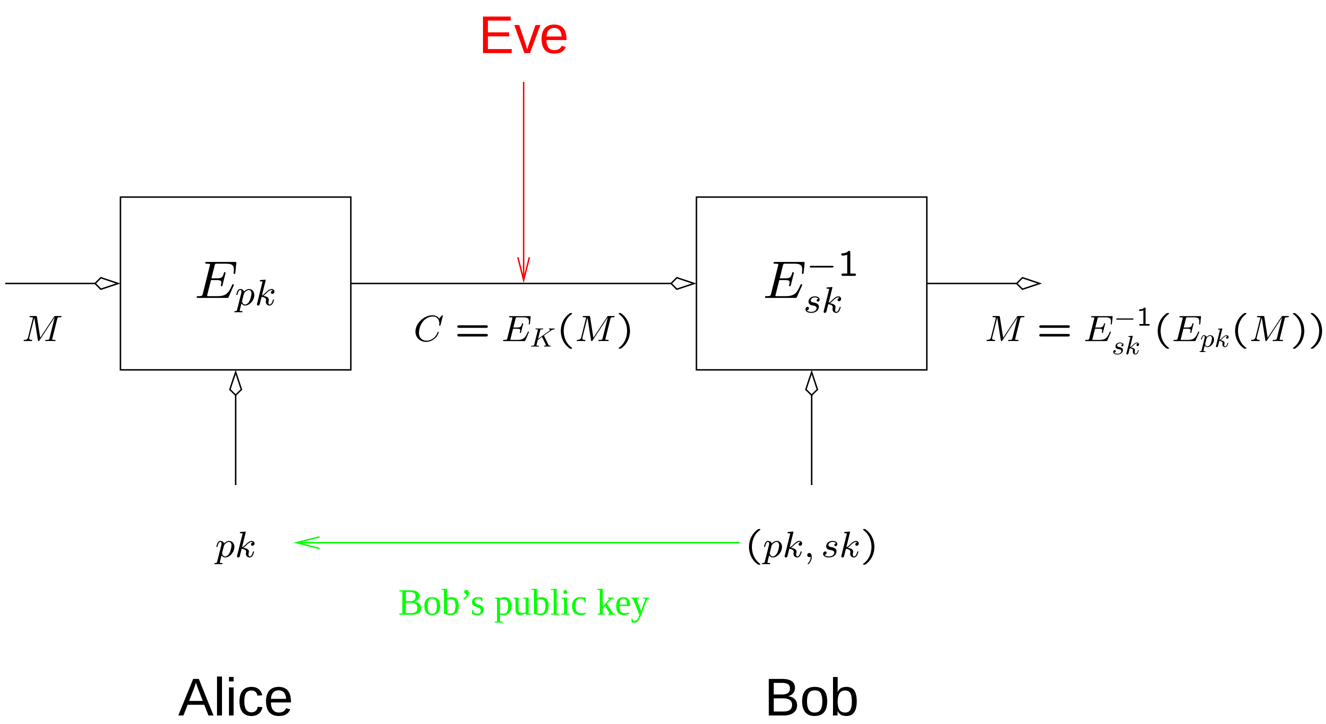
- There were different encryption and decryption functions E and E^{-1}

- We said sharing both is necessary if Alice and Bob want to have bidirectional traffic

- If Alice has a cipher (E, E^{-1}) and Bob has a cipher (F, F^{-1}) then they do not need to share the inverse ciphers!

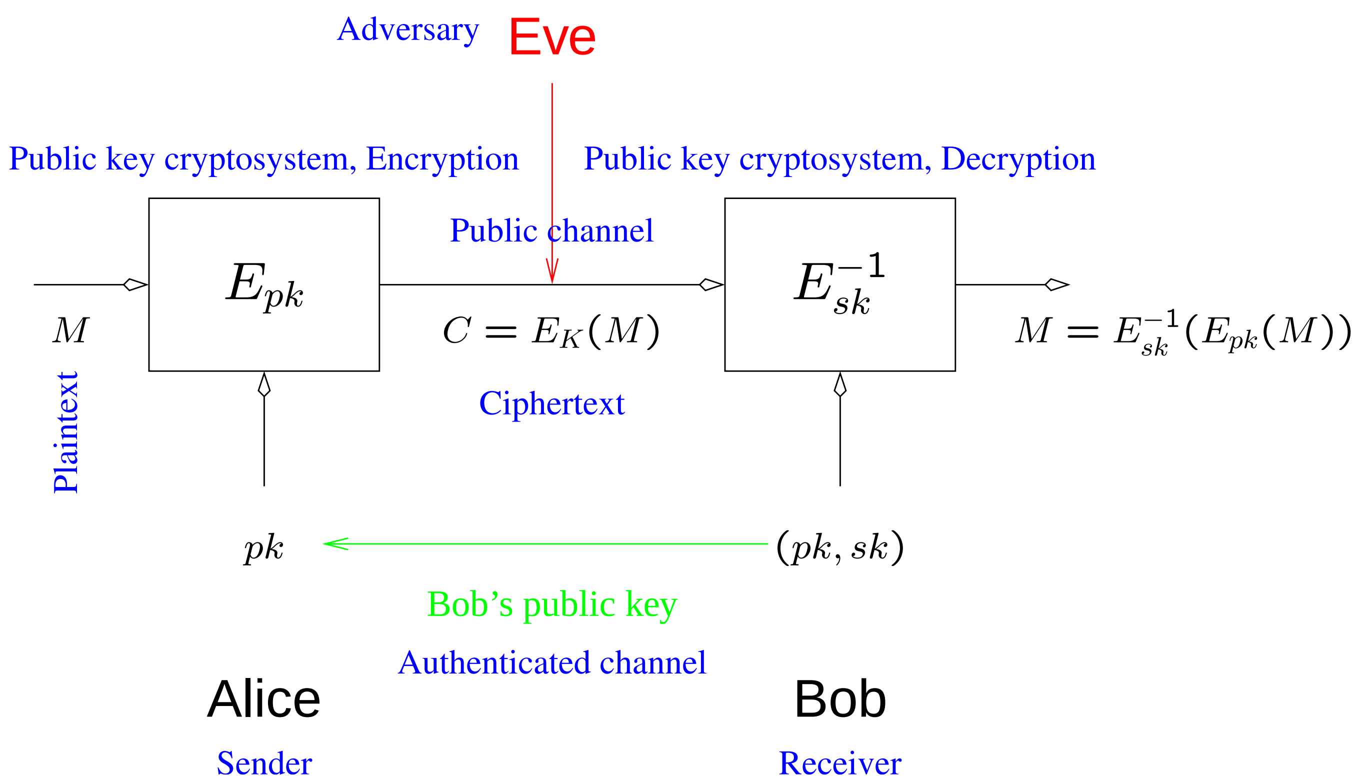
- Recalling the presence of keys, Alice and Bob would not then require to share their respective secret keys

PKC: model



Eve
E_{pk}
M
$C = E_K(M)$
E_{sk}^{-1}
$M = E_{sk}^{-1}(E_{pk}(M))$
pk
(pk, sk)
Bob's public key
Alice
Bob

PKC: model



Alice obtains public key from an *authenticated* channel, no privacy during this necessary!

PKC: Uses

- Secure transmission over an insecure channel
- Secure storage on insecure media
- Authentication
- Digital signatures
- . . .

Why PKC is good?

- In SKC (secret-key crypto) Alice needs a shared secret with everybody else
- In PKC, Alice needs only one secret: her own private key
- Digital signatures provide nonrepudiation
- Many applications (protocols, . . .)

PKC: Uses

- Caveat: public-key cryptography is significantly (up to 1000 times) slower than secret-key cryptography

- Encryption/authentication of long messages is impractical

- Solution for encryption: encrypt messages by using a secret-key encryption scheme with short random key K, and then encrypt K by using a public-key encryption scheme.

- Faster, and requires the storage of encrypted K only

- Authentication: hash the message before signing (see later)

Public-key identification

- Simple idea:

- Alice encrypts a random nonce r by using Bob's public key

- Bob demonstrates the knowledge of his key by sending decrypted r back to Alice

- Other advantage: if somebody tampers Alice's machine, this somebody will not later be able to impersonate Bob

Real protocols are more complicated (hint: malicious Alice)

Digital signatures

- Digital signature algorithm: a function that, given private key d and message M, outputs the signature $C = \mathbf{sign}(d, M)$

- Anybody who has the public key e and M can verify the signature by using a verification algorithm

- Advantage 1: Verifier can obtain e from a central directory after getting the signature

Digital signatures

- Advantage 2: nonrepudiation. In MAC, Alice and Bob share a key K.
- If Alice created $C = \mathbf{MAC}_K(M)$, Bob knows it, but cannot prove it to third parties
- If Alice created $C = \mathbf{sign}(d, M)$, Bob can prove that Alice did it, and make Alice responsible

Hash algorithms

- Keyless algorithms that take an arbitrary long message and compress it into a fixed-length message

- *One-way hash* H: given y, it is hard to compute an M such that $y = H(M)$

- *Collision-intractabe* H: it is hard to find two different messages M and M' such that $H(M) = H(M')$

Password hashing

- If your password M is stored in a open on the server, an intruder can get a copy of it
- Encryption does not help, since you must store the encryption key
- Use one-way hash: store only $H(M)$. Even if intruder gets $H(M)$, she cannot compute M
- Additional benefit: $H(M)$ has fixed length
- Caveat: password file should still be protected to avoid dictionary attacks

Message authentication

- Alice and Bob share a key K, Alice sends M to Bob

- Sending $H(M)$ along with M does not authenticate Alice as M's sender

- Basic idea: compute $H(K, M)$. Shows that you know the key

Comment: this method is not secure, but there are similar secure methods (HMAC)

Message fingerprint

- Alice has a data structure S and wants to check that it has not been tampered
- Solution: store hash $y = H(S)$ in a tamper-proof media, and periodically recompute $H(S)$ and check that it is equal to y
- NB! One must be sure that the program to compute H has not been tampered with

Downline load security

- A device (printer, mobile phone, ...) needs to execute programs but does not have memory to store all of them
- An option is to download them from an external source
- Storing hash of the programs is a possibility of being “sure” you do not execute Trojan horses

Digital Signature Efficiency

- Hash functions are about as efficient as secret-key cryptosystems
- Thus, instead of directly signing a long message, it is practical to hash the message first and then sign the result
- Question: what security requirements should H satisfy here?