



TEKNILLINEN KORKEAKOULU

Basics of Cryptography



What is Cryptography?

- Cryptography is an applied branch of mathematics
- In some situations it can be used to provide
 - Confidentiality
 - Integrity
 - Authentication
 - Authorization
 - Non-repudiation



Simple Encryption Example

- Alice wants to send a message to Bob, without Eve knowing the contents
- Alice and Bob discuss this beforehand and decide to use the *Caesar algorithm* with *key 7*

ABCDEFGHIJKLMNOPQRSTUVWXYZ

TUVWXYZABCDEFGHIJKLMNOPS

- Thus *plaintext* "hello" is "axeeh" in *ciphertext*
- Even if Eve knows the algorithm (Caesar), she needs the key (7) to read the message
 - However a simple exhaustive attack (trying all the different keys) is sufficient to produce the key

```
echo HELLO|tr "ABCDEFGHIJKLMNOPQRSTUVWXYZ" "TUVWXYZABCDEFGHIJKLMNOPS"
```



What We Need to Define to Use Cryptography?

- Participants
- Purpose
 - Encryption, authentication...
- Algorithm(s)
- Key(s)
- Plaintext
- Ciphertext



- Algorithms are the basic building blocks on which a crypto system is built on
- Several types:
 - Symmetric or secret key
 - Asymmetric or public key
 - Hash
 - Key exchange
 - One time pad
- A typical property of most cryptographic algorithms is that when used with a key, they produce a result, which is difficult to replicate without knowing both the plaintext and the key used



- One time pad means using a key only once and typically a key that has as much or more length as the message itself
 - A list of different Caesar keys for each letter
 - A long binary string as long as a message
 - Encryption by XORing to the message
 - Decryption by XORing again
- The sender and recipient must have the same pad
- The pad should be used only once
 - Statistical attacks may be used against re-use of key
- Copying and delivering the one time pads is the largest problem for this kind of system
- With correct key procedures, the algorithm is unbreakable
- Protects **confidentiality**, not integrity



Symmetric Encryption

- Symmetric encryption is based on a secret shared by the participants and on an algorithm
- The secret is used as both the encryption and decryption key
- The symmetric algorithms protect the confidentiality of the data
- Can also protect the integrity when used in combination with other technologies
- Typically the symmetric encryption algorithms are efficient and fast
- The main weakness in symmetric encryption is the need for the *shared secret*
- This produces also a $n(n-1)/2$ type scaling problem in larger installations, unless the system does something special
 - E.g. Kerberos, which has servers trusted by all, thus needing only n shared secrets
- Suomeksi: salaisen avaimen salaus



How Symmetric Encryption Works?

- The main principles are
 - Confusion, bit patterns are substituted for another bit patterns
 - Diffusion, positions of bits are permuted
- Here is a sample of how the IDEA algorithm operates on a block of data, divided to four inputs, using subkeys generated from the encryption key
- This round is repeated several times to produce the encrypted data block

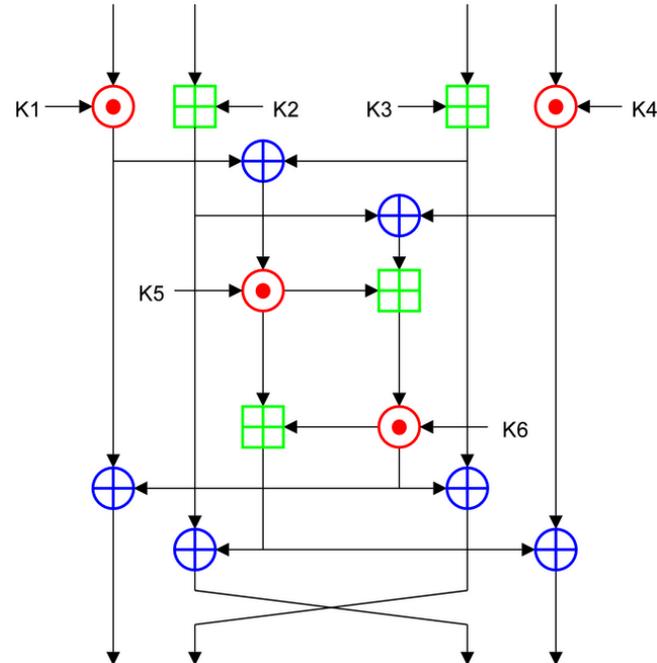


Image © Matt Crypto/Wikipedia

● multiplication modulo $2^{16} + 1$

■ addition modulo 2^{16}

⊕ bitwise XOR



How Block Encryption is Used

- Symmetric Block encryption works on blocks of data (e.g. 64 bits in, 64 bits out)
- Electronic code book means dividing a message to blocks and encrypting each
 - Makes it easy to identify repeated contents (e.g. headers, 64 bits is 8 characters)
 - Usually not used
- Cipher Block Chaining means that a previous cipherblock is XORed to next plaintext block
 - This hides repeated identical plaintext blocks
 - A random *Initiation Vector* is needed to protect the first blocks
- Suomi: lohkosalaus



Stream Encryption

- Symmetric encryption can also operate on a bit or byte stream (one bit in produces one bit out)
 - Suitable for stream data, e.g. voice
- Cipher block chaining encryption can be used to produce a stream of data that is dependent on the key
 - This stream can be used for stream encryption via XOR function
 - Called Output Feedback
- Suomeksi: jonosalaus



Public Key Encryption

- The public key algorithms are based on the properties of several mathematical operations
 - Very roughly: it is easy to multiply two large primes, but difficult to factor the result back to components
- An participant has two keys, related to each other
 - What is encrypted with one key can be opened with the other key
 - One key is called "public" and can be shared with other participants or even made public
 - Another key is called "private" and is kept secret
- A secret message encrypted with the public key can be opened only by applying the private key
- Suomeksi: julkisen avaimen salaus, julkinen ja yksityinen avain



Public Key Encryption

- Public key encryption is usually not very efficient (involves multiple mathematical operations)
- Typically a random session key is created and encrypted with the recipient's public key
 - The session key is used with a symmetric algorithm to encrypt the bulk of data



Public Key Signatures

- Many (not all) public key algorithms have an interesting side effect: the keys can be reversed
- Thus anything encrypted with the private key can only be opened by the public key
 - Which means that it must have been encrypted by the holder of the private key, thus creating a signature



- A hash is a cryptographic one way function that produces a record smaller than the plaintext
 - Sometimes called a fingerprint
- The plaintext can not be recovered from the hash, but it is practically impossible to produce a plaintext that would produce the same hash
- Thus a hash encrypted by the document signer's private key can be used as a signature for a document
- Used to produce Message Authentication Codes (MAC) to verify the integrity of a message
- Hashes are needed to protect message integrity, encryption protects confidentiality
- Suomi: tiivistä



Diffie-Hellman Key Exchange

- When Alice and Bob want to communicate, they can create a shared secret with an easy method that keeps Eve from listening to their communications
 - Alice and Bob agree on using a large prime number p and a base number $1 < g < p$
 - Alice chooses a random number a : $1 < a < p$, and computes $g^a \bmod p$
 - Bob chooses b : $1 < b < p$, and computes $g^b \bmod p$
 - Alice sends Bob $g^a \bmod p$, and Bob sends Alice $g^b \bmod p$
 - Alice computes $(g^b \bmod p)^a \bmod p$
 - Bob computes $(g^a \bmod p)^b \bmod p$
 - Now, Alice and Bob share a common (secret) value, since $(g^b \bmod p)^a \bmod p = g^{ab} \bmod p = (g^a \bmod p)^b \bmod p$ and case it as the basis for encryption



The Meaning of Diffie Hellman

- Eve knows: p , g , $g^a \bmod p$ and $g^b \bmod p$:
 - Determining $g^{ab} \bmod p$ is very difficult
 - Known as the Diffie-Hellman problem
- But Martin can be the man-in-the-middle
- Diffie-Hellman key exchange does not authenticate the participants
- But if Alice and Bob know each other's public keys, they can sign their messages with their private keys, thus providing authentication
- To know their public keys, they need a public key infrastructure (PKI)
- Diffie-Hellman is used by e.g. SSH 1.0 and IPsec



- Symmetric encryption is a relatively lightweight method to protect confidentiality in transit
- Public key cryptography can be used to transmit the session key
- But:
 - How to make sure whom the public key represents?
 - How to authenticate the entity at the other end?



Public Key Infrastructure (PKI)

- Public key cryptography is a powerful tool for verifying an entity's identity
- But managing the keys can lead to a problem of order $n(n-1)/2$
 - Especially if we are supposed to authenticate both participants of communications connections within a large group
- But if there is a trusted participant, public keys can be *certified*
 - *Certificate* binds the key to an entity (person, company)
 - A certificate is signed by encrypting its hash with the *certifier's* private key
- The whole system is called PKI
- Suomeksi: varmenne, varmentaja, julkisen avaimen infrastuktuuri



What Is a Certificate

- A document signed by a party that the system assumes is trusted
- Contains at least:
 - The subject's public key
 - The subject's identity **or** an authorization bound to the key
 - Identity of the signer
 - Signature



Setting Up a PKI

- Alice, Bob and their friends decide to set up a PKI
- They select Trent to be the certifier
- Everybody (including Trent) creates their key pair
- Everybody proves their identity to Trent
 - Trent certifies their public keys
 - Everybody gets Trent's public key
- Now everybody has their private key, certified public key and Trent's public key
- Note that even Eve and Mallory can be certified
- The only part that has to be secret is the private keys



How Is a PKI Used?

- Alice contacts Bob
 - Bob does not know who he is talking to
- Alice and Bob do an initial Diffie-Hellman key exchange
 - Eve does not know who is calling Bob
- Alice presents her certificate to Bob
 - Bob gets Alice's public key
 - Bob has Trent's public key and verifies the certificate signature
 - Bob is still not sure he is talking to Alice
- Bob sends Alice a random number
- Alice multiplies it with another random number
- Alice encrypts the result with her private key and sends it to Bob, with the random number
- Bob decrypts the message, verifies the number by multiplying his with Alice's and comparing to the message
 - Now Bob knows he is talking to Alice



What is this Thing about Multiplying Random Numbers?

- Cryptography is very hard to do right and there are many pitfalls
- If Alice would just encrypt Bob's number with her private key Bob could have a contract where Alice gives him her car and send the hash of the contract to Alice
 - Thus Alice generates the random number and modifies Bob's data with it
- These random numbers should be used only once, they are called *nonces*
- BTW, note that in the previous example Mallory could easily do a man-in-the-middle attack



The Benefits and Faults of a PKI

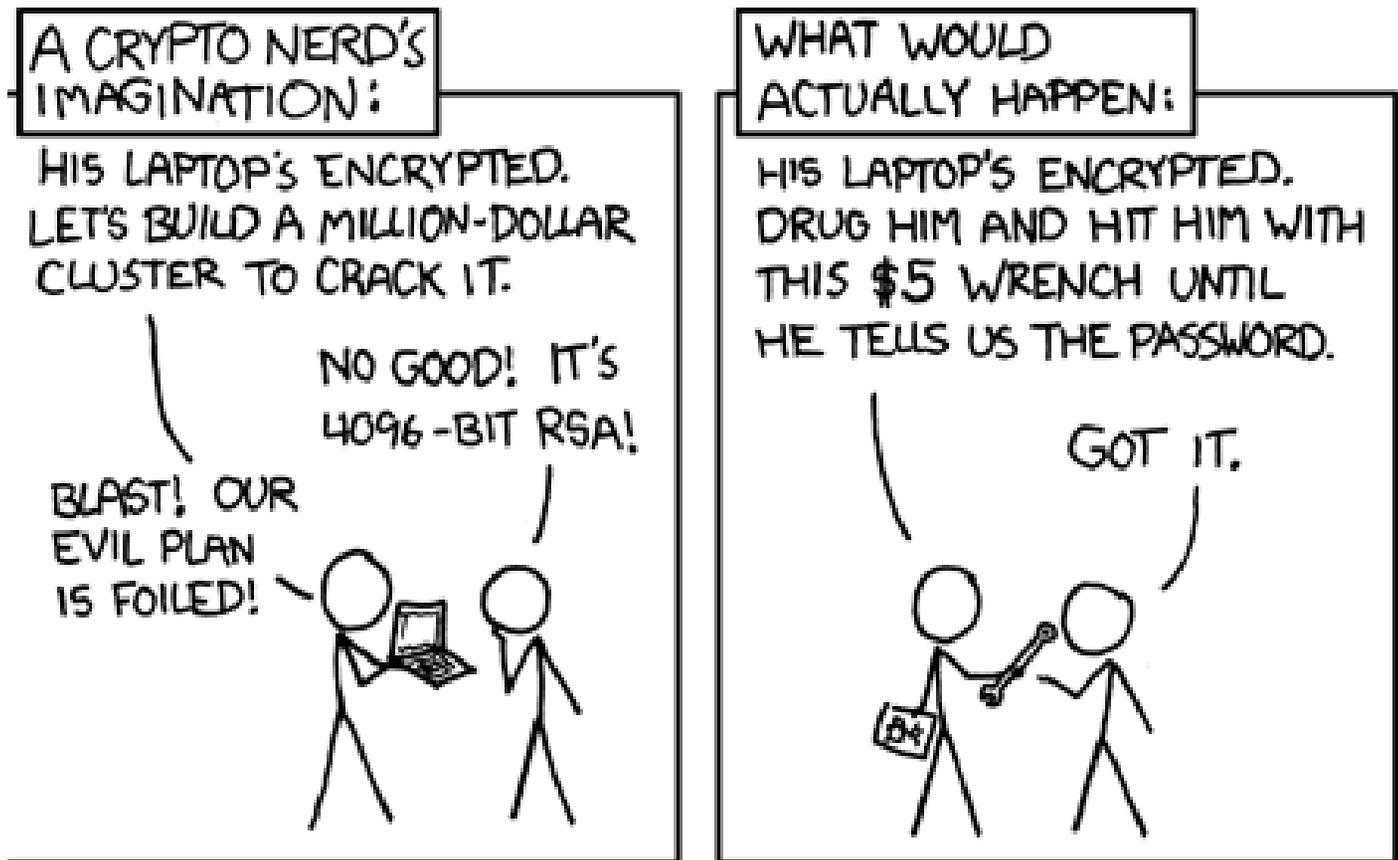
- **Benefits:**
 - Only N authentications needed for N participants
 - Certifier can keep his private key off-line
 - No need for online certificate creation
 - No need for anybody to reveal their private key
- **Faults:**
 - If the certified is untrustworthy, he can create false certificates
 - A complex and difficult to understand system
 - Requires online components (discussed later)
 - Selecting the certifier is not easy



- Cryptanalysis is the science and art of breaking algorithms and cipher messages
- Relies much on statistical methods and analysis of data patterns on the ciphertext
- Several attack models
 - Ciphertext only
 - Known plaintext and ciphertext
 - Chosen plaintext and ciphertext
- Brute force attack of going through the whole keyspace is utilized if keys are short enough
 - 128 meaningful bits is currently too much and 256 bits impossible
 - Note that some algorithms use keys where there is redundancy, e.g. 512 bit RSA key is not considered secure
- Current algorithms can be considered unbreakable, but cryptanalysis is also valuable as a method of evaluating algorithms



Rubber-hose Cryptanalysis





- Whole systems can be created from these primitives
- A system requires usually that several algorithms are combined with key management to do something practically useful
- Here PGP and SSH are presented as examples of working systems



PGP (Pretty Good Privacy)

- Designed by Phil Zimmermann for providing cryptographic protection of e-mail and file storage
 - Uses strong cryptographic algorithms (for its own time, published 1991)
- Offers
 - Authentication using digital signatures
 - Confidentiality with the use of encryption
- Technical features
 - Byte conversion to ASCII for e-mail
 - Key management uses e-mail addresses as subject labels



PGP Design Philosophy

- Written for individual, technically skilled end-users
 - Every user creates and manages their own keys
 - Every user has a freedom to choose, whom to trust
 - No administrative organization or governments involved in operation
 - No hierarchy in trust relationships
- Independently produced, no standardization organizations involved
 - Original versions open source, free of charge
 - Later commercialized and several incompatible versions exist



PGP Key signing

- It provides a mechanism for signing other users keys
- PGP was designed before certificates become popular
 - Only real difference is the lack of data structure, effect is similar to a PKI
- PGP certification structure has no root or entity trusted by everybody
- Anybody can sign anybody's keys and anybody can select whose signatures to trust
 - A web of trust instead of a tree



PGP Key Signing

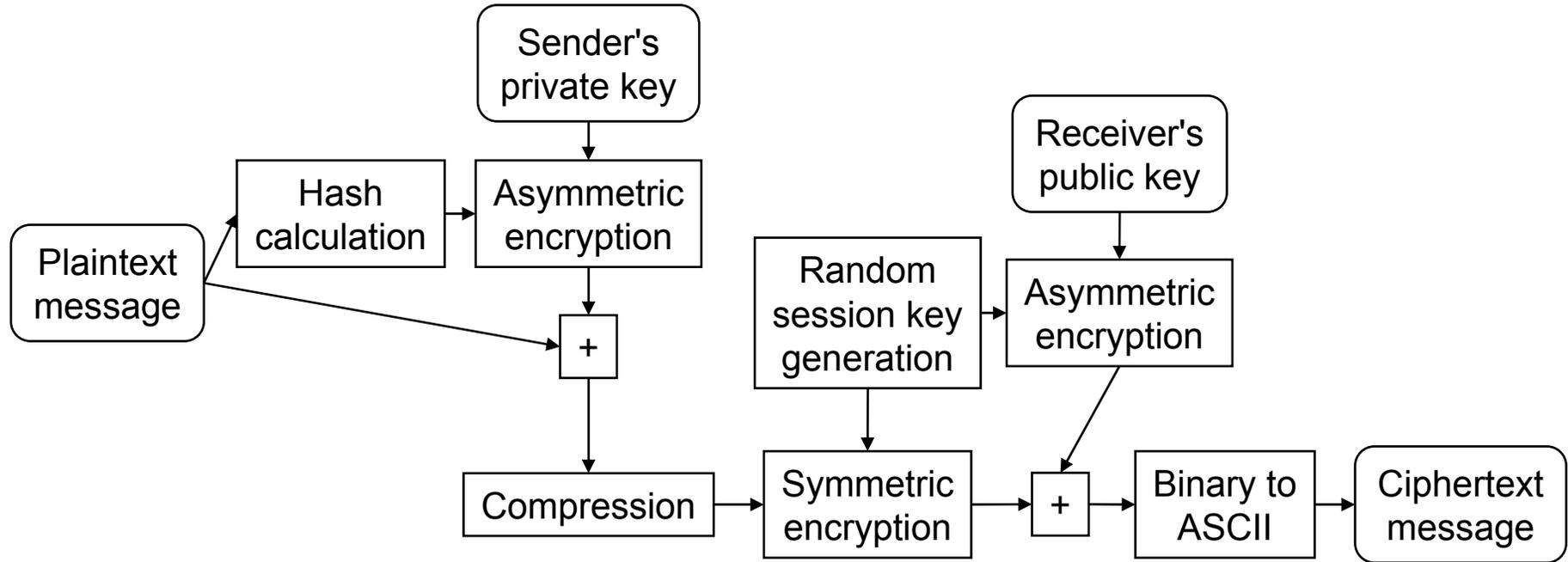


- <http://xkcd.com/364/>
- PGP users can sign other users' keys
 - Forming a web of trust



Putting It All Together

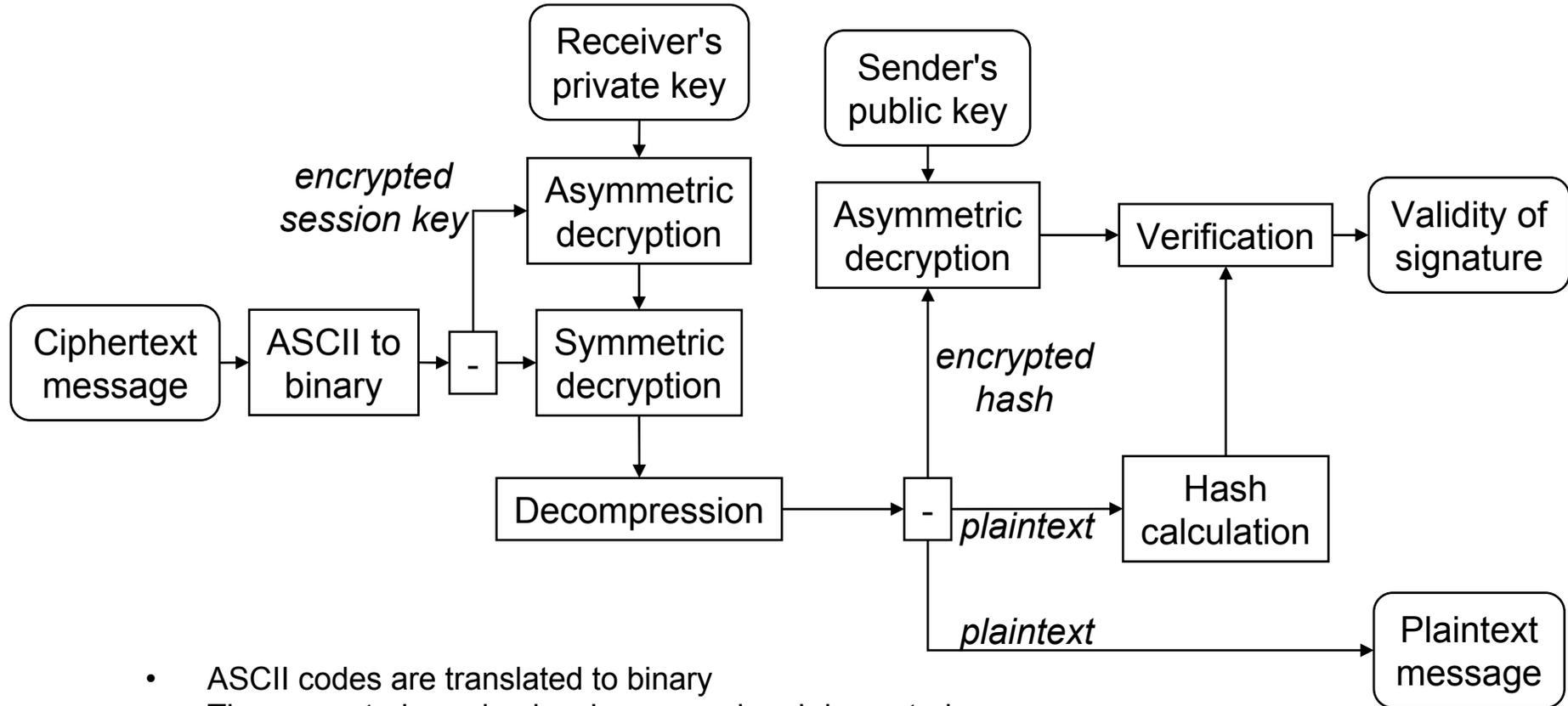
Sending a PGP Message



- The message is signed, compressed and encrypted
- The encrypted session key is added to the end of message
- Binary message is translated to characters, which pass through the e-mail system



Receiving a PGP Message



- ASCII codes are translated to binary
- The encrypted session key is removed and decrypted
- The message is decrypted and decompressed
- The encoded hash is removed and decrypted
- The hash is recalculated and compared to the hash in the message



- SSH (Secure SHell) provides an encrypted TCP connection between two hosts on the network
 - Replaces Berkeley R-tools (rlogin, rcp, rsh)
 - Protects X-Window system traffic
 - Any TCP-connection can be tunneled over SSH
- Vulnerable to “Man in the Middle” -attack
 - SSH client does not know the host key until first connection
- Used for
 - Terminal connections
 - File copying (scp)
 - Tunneling traffic from external hosts to the protected domain
 - E.g. X Window System, CVS, rsync, e-mail

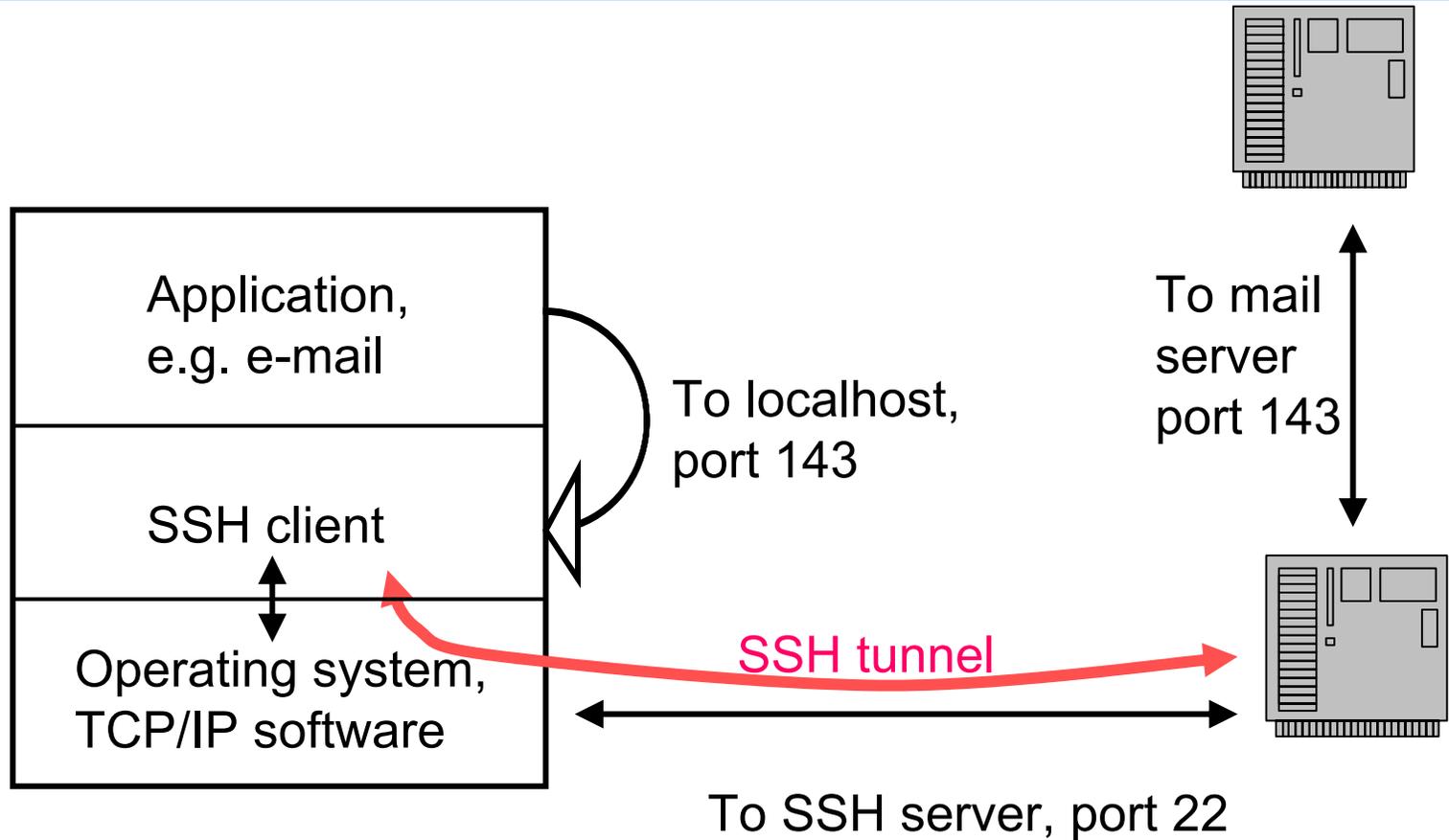


TCP Tunneling

- A local SSH client can be configured to tunnel TCP connections from a local TCP port to a SSH server host and from there to another host
- This protects the traffic between the two SSH hosts
- Use requires changes to software settings, local host appears to be the server
- SSH server is usually in port 22



Tunneling with SSH





SSH 2.0 Protocol Key Exchange

- Client contacts a server (a TCP connection is initiated)
- Server sends two public keys (server and host) and available algorithms
- Client simultaneously sends available algorithms
- Client creates a session key (symmetric), encrypts it with server's public keys and sends it to server
- A shared secret is now formed and a session is started
- Either side may request a renegotiation of keys
- User authentication is done after this



Some Fingerprints for Homework

MD5 Fingerprint:

30:FD:5B:AB:03:8D:A1:DB:E2:E2:6F:C4:F0:ED:AA:37

SHA1 Fingerprint:

19:5A:9F:C7:67:94:9B:D0:15:F1:70:8C:FC:BE:DA:C0:21:F0:7E:0D



Possible Exam Questions

- If you have symmetric, asymmetric and hash cryptographic functions (6p each question)
 - How would you send a message while protecting its confidentiality and ensuring that the recipient can verify it is from you?
 - What additional information do you need (e.g. public and private keys)?
 - How would you protect a connection from eavesdropping and tampering (remembering the PGP diagram is fine, but not enough)?
 - How would you design a system for government announcements in such a way, that the public can verify the authenticity of the messages?
 - If this examination were done on a computer, design a system that prevents a malicious assistant from changing your exam afterwards while making it hard for you to claim that a document written afterwards is your exam.
- Reduce the requirements to the CIA fundamentals, think are we sending messages or protecting connections, work on the secure side (over-engineering is not a problem)



- T/F: Symmetric crypto is not really needed after the invention of public key crypto (1 p)
 - CPU power
- What is the difference between one time pad and one time password? (2 p)
- Where is block cipher usable and where is stream cipher preferred, give one sample of each case. (2 p)
 - Block w/ XOR is stream, digital data, digitalized analog data. Expand this to about 6-10 lines and a possible diagram or if you really know your stuff, add a few words and reply in two lines.



- Why are CLR's needed with a PKI?
 - A private key is revealed -> anybody can sign/be authenticated
- What is the purpose of a certificate?
 - Digitally signed document with public key and tons of cool stuff that can be used to represent the entity who has the corresponding private key
- If you receive a certificate and you have and trust the certificate issuer's public key, what do you know about the entity giving you the certificate?
- How do you verify the identity of an entity presenting you with a certificate?



- How are nonces used?
 - data that the sender knows is random, but the recipient assumes can be an attack and never crypts/decrypts by itself
 - Make a session unique
- Define: nonce, certificate
- Design a system for signing and storing medical recipes on a smartcard, so that a customer of the national health system can take the recipe to pharmacy and purchase medication. The participants are doctor, patient, pharmacy and national health authority. (6 p)
 - A hard one, analyze each participant and figure out that the doctor needs to sign a recipe and the pharmacy needs to be able to verify the doctor's signature