Introduction to Cryptography
(or 2000 years of Crypto in 2 hours)

Fabio Massacci

Agenda

- Topic
  - Introduction
  - Symmetric key cryptography
  - Hash (one-way) functions
  - Public key cryptography
  - Digital signatures
  - Authentication – key establishment
- Disclaimer
  - This is a quick summary to provide an introduction to people who have no crypto background. For the real thing attend the Crypto course.

Terminology

- Cryptography
  - the science of designing methods “secret writing”.
- Cryptanalysis
  - The science of methods for analysing and breaking ciphers.
- Cryptology = cryptography & cryptanalysis.
- Cryptography today
  - the study of mathematical techniques related to aspects of information security, such as confidentiality, data integrity, entity authentication, and data origin authentication.
- Why do we need it?

Why do we need cryptogrpahy?

- Because we want to talk over a channel that only process bits
- Do bits have colors?
  - Alice sends Bob a stream of “green” bits b1…bn
  - Charlie sends Bob the same stream of “red” bits.
  - If bits had colors Alice could tell them apart
- Are bits invisible?
  - Alice sends Bob a stream of “invisible” bits b1..bn
  - Charlie can read the stream b1…bn
  - If bits were invisible only Bob could read them

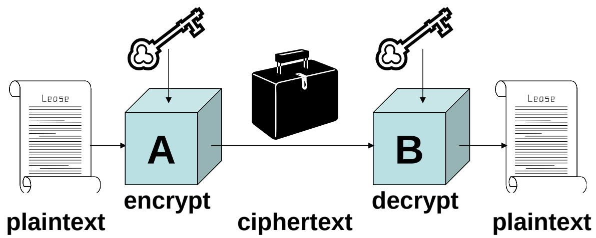
Security services

- Data confidentiality: encryption hides the content of messages.
- Data integrity: integrity check functions (hash functions) detect changes to documents.
- Data origin authentication: digital signatures and message authentication codes verify the source and integrity of documents
- More services may mean authentication against a third party vs authentication for yourself only

Encryption

- Encryption algorithms (ciphers) protect the confidentiality of data.
  - Some (but not all) encryption algorithms can also be used for integrity checks.
- A plaintext (clear text) *x* is converted into a ciphertext *eK(x)* under the control of a key *K.*
- Decryption with an appropriate key **K'** computes the plaintext from the ciphertext *dK'(eK(x))=x*
- Properties
  - If you don't know K' the message should look random.
  - Relation between K and K' determines type of crypto

Symmetric key encryption



Symmetric Key Cryptography

- Properties
  - Symmetric ciphers (secret key cryptography): same key used for encryption & decryption.
  - Encryption protects documents on the way fromAtoB.
  - A and B have to share a key and keep their keys secret.
  - A procedure is required for A and B to obtain their shared key.
  - For n parties to communicate directly, about n2keys are needed.
- Example
  - SWIFT (the network used for international bank transfers) use symmetric keys to encrypt data in transit
  - Long ago people with a suitcase full of key material (a tape) had to physically bring the key material across the world.

Block ciphers & stream ciphers

- Block ciphers: encrypt sequences of “long” data blocks without changing the key.
  - Security relies on design of encryption function.
  - Typical block length: 64 bits, 128 bits.
- Stream ciphers: encrypt sequences of “short” data blocks under a changing key stream.
  - Security relies on design of key stream generator.
  - Encryption can be quite simple, e.g. XOR.
  - Typical block length: 1 bit, 1 byte, 8-bit word
- Typical usage
  - Stream cipher → streaming data (eg phone conversation)
  - Block cipher → data at rest (eg image)

One time pad

- Very simple algorithm
  - Given a streaming sequence of N message bits p_i
  - Take a sequence of N truly random bits k_i
  - To encrypt → c_i=k_i xor m_i
  - Decrypt → m_i = m_i xor k_1
- Properties
  - Perfect confidentiality in information theoretic sense
    - BUT only if you use the key only ONCE
  - Zero integrity protection
    - (so good only if integrity protected otherwise)
  - Hugely expensive: truly random sequence are difficult to generate

Sorry man, I have a 1GB file…

- If your file is large and your block ciphers only do 128B how do you do encrypt the whole?
  - “chain” the result of the first encryption to encrypt the data of the second and so on
  - Basically generalize the idea of the one-time pad
- Key-only chaining (stream ciphers style)
  - K_i+1 = enc(K,K_i)
- Text and Key chaining (many variants)
  - C_i+1 = P_i+1 xor enc(K,C_i)
- Advantage:
  - if touch a bit decryption fails → can spot manipulation
- Disadvantage:
  - if you touch a bit decryption fails → can’t recover the plaintex

Block Ciphers

- Algorithms: AES (Rijndael), DES, 3DES, …
- Typical block sizes: 64, 128, 256 bits.
- No provable security.
- Algorithms designed to resist known attacks: e.g. differential & linear cryptanalysis.
- Recommended key length: 80-90 bits.
- DES: 56-bit keys vulnerable to brute-force key search.

Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	2^{31} μs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} μs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} μs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} μs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} μs = 6.4×10^{12} years	6.4×10^6 years

Cryptographic hash functions

- Cryptographic hash functions are used for integrity checks.
- Apply a hash function *h* to a document *x* and store the result *h(x)* in a secure place.
- The result *h(x)* is called "hash value", "message digest", or "checksum".
- Changes to *x* detected by re-computing the hash of *x* and comparing the result with the stored value.

Security properties

- Ease of computation: it is easy to compute *h(x).*
- Compression: the hash function maps inputs of arbitrary length to fixed length results.
- Pre-image resistance (one-way): given *y*, it is computationally infeasible to find *x* so that *h(x)=y.*
- More properties
  - weak collision resistance
    - computationally infeasible given x to find $y \neq x$ such that H(*y*) = H(*x*)
  - strong collision resistance
    - computationally infeasible to find any pair (*x*, *y*) such that H(*x*) = H(*y*)
- Many hash functions: SHA,RIPEMD-160, MD5 (dubious)
- *As of now no proof that they always work*
  - *Actually their existence is at the core of P vs NP question*

Message authentication codes

- Hash functions do not need keys.
- For integrity checks, hash values have to be protected: keys for cryptographic protection.
- Message authentication code (MAC): keyed hash function for data origin authentication.
- HMAC construction: take a hash function *h*, for a key *k*and a document *x*, compute $HMAC(x) = h(k||p_1||h(k||p_2||x)$
  (p_1, p_2 are padding fields, || is concatentation)

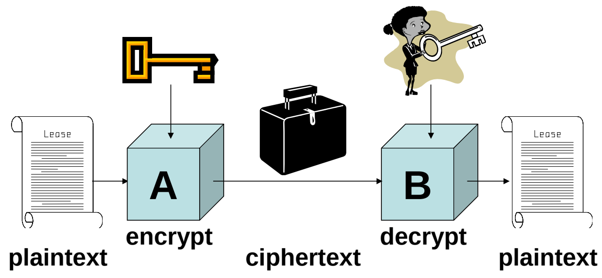
Back to square one?

- Symmetric key encryption:
  - sender and receiver have to share a key.
- Keyed Hash:
  - sender and receiver have to share a key
- To send a secret message from *A* to *B*, or to verify the integrity of a message from *A* to *B* we have to get a secret key to *A* and *B* first.
  - To solve the problem of sending secret messages, we have to solve the problem of sending secret keys.
- Are we moving in circles?

Public key encryption

- Proposed in the open literature by Diffie & Hellman in 1976.
  - Arguably invented by British Secret Service some year earlier
- Each party has a public encryption key and a private decryption key.
- Computing the private key from the public key should be computationally infeasible.
- The public key need not be kept secret but it is not necessarily known to everyone.
- There exist applications where access to public keys is restricted.

Encryption with public keys



Basic idea

Protocol for A

- A and B share off line
  - common parameters Pab
- A computes
  - random number Xa
- A sends
  - 1-way-A(Pab,Xa)
- A computes
  - Combine-A(1-way-B(Pab,Xb),Xa)
- If the math commutes A and B share a secret

Protocol for B

- A and B share off line
  - common parameters Pab
- B computes
  - random number Xb
- B sends
  - 1-way-B(Pab,Xb)
- B computes
  - Combine-B(1-way-A(Pab,Xa),Xb)
- If the math commutes A and B share a secret

Diffie Hellman

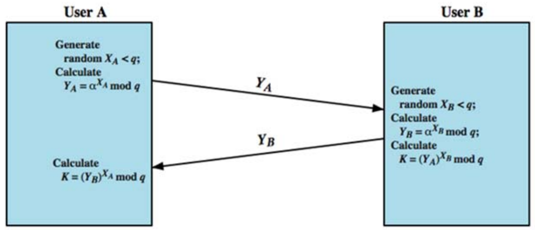
- Based on the discrete log problem
- Given
  - p is prime number
  - g is a primitive root of p (generator)
    - powers of g generate all integers from 1 to p-1
  - b any integer < p
- Compute discrete logarithm a
  - $b = g^a \bmod p$ for some a. $0 \leq a \leq p-1$
- Computationally untractable (as of now)
- However something commutes
  - $(g^a)^b=(g^b)^a$

Diffie-Hellman Example

- Have
  - prime number q = 353
  - primitive root g = 3
- A and B each compute their private secrete (Xa,Xb) and public keys (Ya,Yb)
  - A computes Ya = 3^{97}mod 353 = 397 mod 353 = 40
  - B computes Yb = 3^{233}mod 353 = 3233 mod 353 = 248
- then exchange (Ya,Yb) and compute shared secret
  - for A: K = $(Yb)^{Xa}$ mod 353 = 24897 mod 353 = 160
  - for B: K = $(Ya)^{Xb}$ mod 353 = 40233 mod 353 = 160
  - Math (modulo p): $(Yb)^{Xa} = (g^{Xb})^{Xa} = g^{Xb*Xa} = (g^{Xb})^{Xb} = (Ya)^{Xb}$
- attacker must
  - Solve 3^X mod 353 = 40 to find Y which is hard
  - desired answer is 97, then compute key as B does

Key Exchange Protocols

- Picture from Stalling's Book:



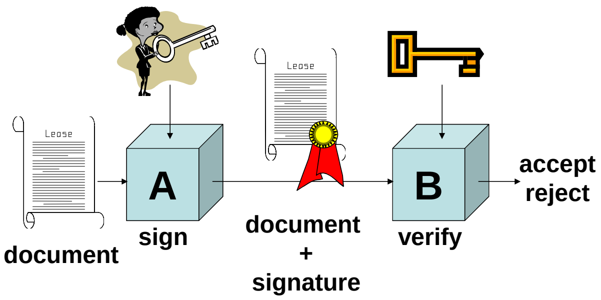
Some complications…

- What if messages can be changed in transit?
- How do you beat a Chess Master?
  - Have another chess master playing against him
- DH can be attacked because it only guarantee confidentiality but not integrity
- Attack
  - A send YA to B and C intercepts it
  - C sends YC to B claiming to be A
  - B sends YB to A and C intercepts it
  - C sends YC to B claiming to be B
  - What has A? What has B? What has C?

Digital signature mechanisms

- Used for non-repudiation, origin authentication and data integrity services.
- Used in some authentication exchange mechanisms.
- Digital signature mechanisms have three components:
  - key generation
  - signing procedure (private)
  - verification procedure (public)

Digital signatures



Digital Signatures

- *A* has a public verification key and a private signature key(→ public key cryptography).
- *A* uses her private key to compute her signature on document *m*.
- *B* uses a public verification key to check the signature on a document *m* he receives.
- At this technical level, digital signatures are a cryptographic mechanism for associating documents with verification keys.

Digital Signatures continued

- To get an authentication service that links a document to *A*'s name (identity) and not just a verification key, we require a procedure for *B* to get an authentic copy of *A*'spublic key.
- Only then do we have a service that proves the authenticity of documents 'signed by *A*'.
- Yet even such a service does not provide non-repudiation at the level of persons.

Electronic signatures

- Digital signatures: mathematical evidence linking a document to a public key.
- Electronic signatures: a security service for associating documents with legal persons.
- The link between a public key and a person has to be established by procedural means.
- This link can be recorded in a certificate.
- Certificates are not necessary for verifying digital signatures, verification keys are.

Certificates

- How do you get a verification key?
  - Public key cryptosystems often assume there is a public directory of user names and keys.
  - But a “Global PK directory” does not exists
- Kohnfelder [1978]: implement the directory as a set of digitally signed data records containing a name and a public key; he coined the term certificate for these records.
- Today: a certificate is a signed document binding a subject to other information; subjects can be people, keys, names, …

30/09/2014 Massacci - Paci - Security Engineering 29

Certification Authorities

- Certificates are signed by an Issuer.
- Certification Authority (CA) is just another name for Issuer.
- Sometimes CA is used more narrowly for organizations issuing ID certificates [PKIX].
- The application determines the technical and procedural ‘trust’ requirements a CA has to meet.
- Sometimes Trusted Third Party (TTP) is used as a synonym for CA.

30/09/2014 Massacci - Paci - Security Engineering 30

Public Key Infrastructures

- The protocols, services and standards that facilitate the use of public-key cryptography by allowing the secure distribution of public keys between communicating parties.
- PKI standards: X.509 [ISO/IEC 9594-8], PKIX [RFC 2459], PKCS.
- X.509 certificates were intended to bind public keys [originally passwords] to X.500 path names (Distinguished Names) who has permission to modify X.500 directory nodes.

30/09/2014 Massacci - Paci - Security Engineering 31

X.509 certificates

- User certificate (public key certificate, certificate): the public key of a user, together with some information, rendered unforgeable by encipherment with the secret key of the certification authority which issued it.
- Attribute certificate: a set of attributes of a user together with some other information, digitally signed under the private key of the CA.
- Certification authority: an authority *trusted* by one or more users to create and assign certificates.

30/09/2014 Massacci - Paci - Security Engineering 32

X.509v3 certificate format

- version (v3)
- serial number
- signature algorithm id
- issuer name
- validity period
- subject name
- subject public key info
- issuer unique identifier
- subject unique identifier
- extensions
  - extensionID
  - critical: YES/NO
  - extensionValue

Extensions: added to increase flexibility.

Critical extensions: if a critical extension cannot be processed, the certificate must be rejected.

Critical extensions are also used to standardize policy.

Revocation

- Certificates may have to be revoked
  - if a corresponding private key is compromised.
  - if a fact the certificate vouches for no longer is valid.
- Certification Revocation Lists (CRLs):
  - Distributed in regular intervals or on demand.
  - Make sense if on-line checks are not possible or too expensive.
- When on-line checks are feasible, certificate status can be queried on-line:
  - Online Certificate Status Protocol – OCSP.
  - Positive lists in the German signature infrastructure.

Question….

- In order to verify a certificate you need a verification key.
- If you get a verification key from a certificate, how do you get a certificate?
- Ask a certificate server to send you a signed certificate
- How do you get the verification key to check the key that signed the certificate?
- Ask a certificate server to send you a signed certificate certifying the certificate server…
- Spot a certain loop here?
- Need a root of trust
- How many people trust

How to bootstrap trust?

- Do you trust…
  - TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hizmetleri A.Ş.
  - A-Trust Ges. f. Sicherheitssysteme im elektr. Datenverkehr GmbH
  - Go Daddy Root Certificate Authority - G2
  - CA 沃通根证书
  - XRamp Security Services Inc
- How many of you actually trusted in the last month…
  - GeoTrust Universal CA

Oh man but I have still my 1GB…

- How do you sign a 1GB file?
- The mathematician's correct answer
  - I just use elliptic curve over a field of "1GB" instead of 512Bits
  - You'll die before the algorithm calculates that
- The computer scientist's hack
  - Let m be very large
  - Compute hash(m) – now this is small
  - Compute sign(privK,hash(m))
  - If hash is broken → digital signature down the pipe

Sorry man, I also need encrypting my 1GB

- How do you encrypt a 1GB file?
- The dumb CS hack
  - Let m be very large
  - Compute hash(m) – now this is small
  - Compute enc(privK,hash(m))
- The smart CS's hack
  - Let m be very large
  - Generate a random symmetric key sessionK
  - Compute enc(sessionK,m), enc(pubK,sessionK)
  - Send both
  - If symmetric encryption broken → security broken

Key exchange and distribution

- Crypto transforms (communications) security problems into key management problems.
- To use encryption, digital signatures, or MACs, the parties involved have to hold the "right" cryptographic keys.
- With public key algorithms, parties need authentic public keys.
- With symmetric key algorithms, parties need shared secret keys.

Key usage

- It is good cryptographic practice to restrict the use of keys to a specific purpose.
- In key management, we may use key encrypting keys and data encrypting keys.
- Examples for key usages:

  Encryption Decryption

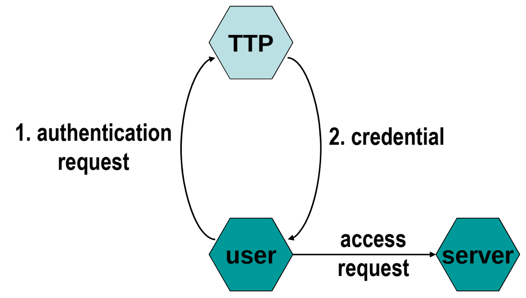
  Signature Non-repudiation

  Master key Transaction key …
- With RSA, don't use a single key pair both for encryption and for digital signatures.

Key establishment & TTPs

- In a protocol like TLS where key authentication is based on digital signatures, we may need a Trusted Third Party (TTP) to vouch for the authenticity of verification keys.
- In a protocol where authentication is based on symmetric cryptographic algorithms, a TTP may serve as a key distribution centre (KDC) supplying parties with session keys.

Key distribution



Key distribution

- Advantage: scalability – a single authentication server can support many servers and users.
- The credential tells the server that the user has been authenticated.
- The credential could contain a session key to be shared between user and server so that the server can authenticate further requests.
- The TTP would also serve as Key Distribution Centre (KDC).

Kerberos

- Kerberos was developed at MIT for user authentication in a distributed system.
- The parties involved are client *A*, server *B*, and Kerberos authentication server (KAS) *S*.
- Based on the Needham-Schroeder key establishment (“authentication”)protocol: the server provides A and B with a session key.
- Uses a symmetric encryption algorithm.
- More of this in the network security lectures

Cryptographic keys

UNIVERSITY OF TRENTO

- Most cryptographic algorithms take a key as one of their inputs.
- Kerckhoffs' principle: Do not rely on the secrecy of cryptographic algorithms; only the keys have to be kept secret.
- State of the art: Standardized algorithms that have been examined quite intensively and are often the strongest part in a security architecture.
- Good key management practices are required to reap the benefits of strong cryptography.

30/09/2014 Massacci - Paci - Security Engineering 45

Key management questions

UNIVERSITY OF TRENTO

- Where are keys generated?
- How are keys generated?
- Where are keys stored?
- How do they get there?
- Where are the keys actually used?
- How are keys revoked and replaced?

- When keys are stored on a computer, cryptography relies on strong computer security.
- If the keys are computer generated (and not truly random numbers) cryptography rely on strong random number generators

30/09/2014 Massacci - Paci - Security Engineering 46

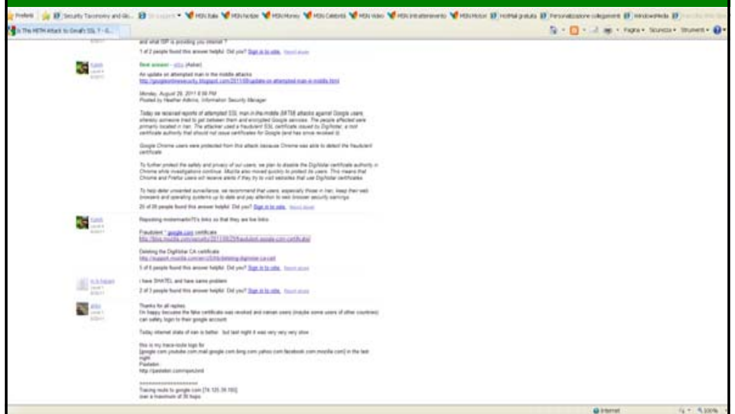
The DigiNotar CA Break

UNIVERSITY OF TRENTO

30/09/2014 Massacci - Paci - Security Engineering 47

The DigiNotar CA Break

UNIVERSITY OF TRENTO



30/09/2014 Massacci - Paci - Security Engineering 48

Resources

- Chapters 2,19, 20 and 21. Stallings & Brown. Computer Security Principles and Practice.
- Chapter 14,15. Dieter Gollmann. Computer Security
- Alfred Menezes, Paul van Oorschot, Scott Vanstone: Handbook of Applied Cryptography http://www.cacr.math.uwaterloo.ca/hac/
- Bruce Schneier: Applied Cryptography