ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Crypto basics

COM-402: Information Security and Privacy

Outline

- **Introduction**
- Symmetric crypto
- Cryptographic hash functions
- Data Integrity
- Authenticated encryption
- Public Key crypto
- Key distribution
- Public Key Infrastructure
- TLS
- HTTPS

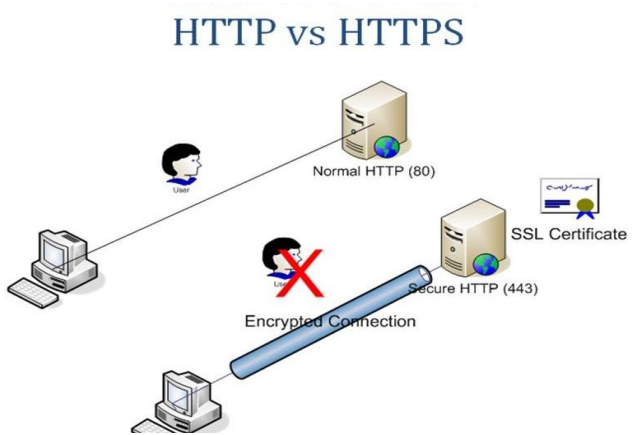
Introduction

- What is cryptography?
  - A toolbox for many security mechanisms
  - Information security and communication security
  - Secure data at rest and data in motion
- Cryptography is not:
  - The solution to all security problems
  - Reliable unless implemented properly
  - Reliable unless used properly
  - Something you should try to invent or implement yourself

Note: Slide borrowed from https://crypto.stanford.edu/cs155/lectures/07-crypto.pdf

Crypto is Everywhere

- Secure communication
  - HTTPS
  - WEP/WPS
  - SSH
  - GSM
  - …



Crypto is Everywhere

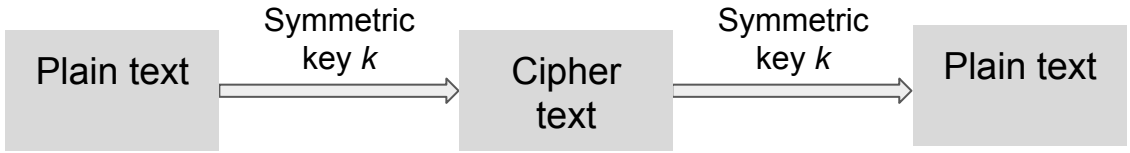
- File and disk encryption:
  - VeraCrypt
  - BitLocker
  - GPG
  - …
- Authentication of users (Alice) and servers (gmail.com)
- Digital cash
- Digital Rights Management
- ...

What we want to do with Crypto

- Confidentiality - Only users with appropriate key can access information
- Integrity - ensure that data is not modified by unauthorized parties
- Authentication - provide a way to identify the author of information
- Accountability
- Availability
- Timestamping
- ...

Two Types of Crypto

- Symmetric or private-key crypto - One shared key



- Asymmetric or public-key crypto - A pair of public and private key

Brief History

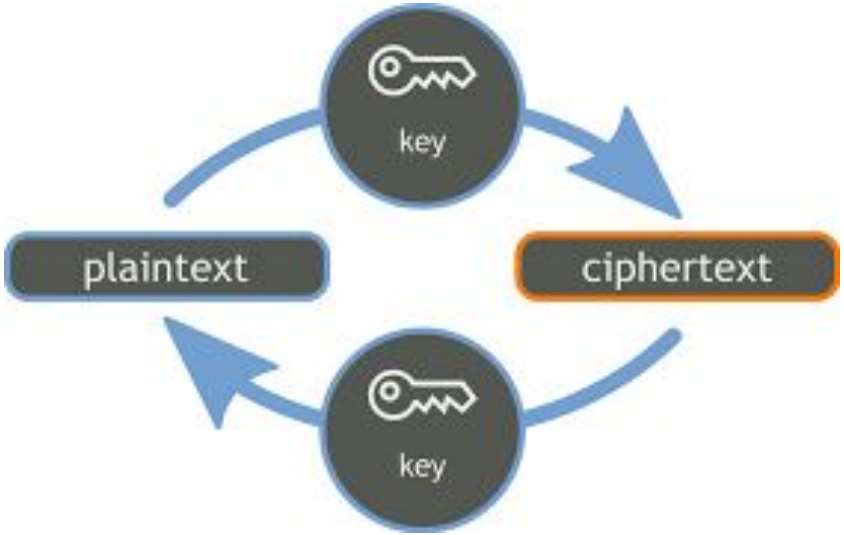
- 50BC - Caesar's Cipher - substitution
- 1883. Kerckhoffs' Principle – A cryptosystem should be secure even if everything about the system, except the key, is public knowledge
- 1917./1918. One-time pad
- Before and during the WW II - Enigma machine
- 70s - Data Encryption Standard (DES)
- 70s - Diffie-Hellman Public Key Crypto
- 70s - Rivest, Shamir, Adleman (RSA)
- 90s - Digital Signature Algorithm (DSA)
- 90s - Secure Hash Algorithm (SHA-1)
- 2001 - SHA-2
- 2001 - Advanced Encryption Standard (AES)
- 2015 - SHA-3

Outline

- Introduction
- **Symmetric crypto**
- Cryptographic hash functions
- Data Integrity
- Authenticated encryption
- Public Key crypto
- Key distribution
- Public Key Infrastructure
- TLS
- HTTPS

Symmetric crypto - Definition

- Encryption of plaintext and decryption of ciphertext are done using the same key, hence *symmetric* crypto

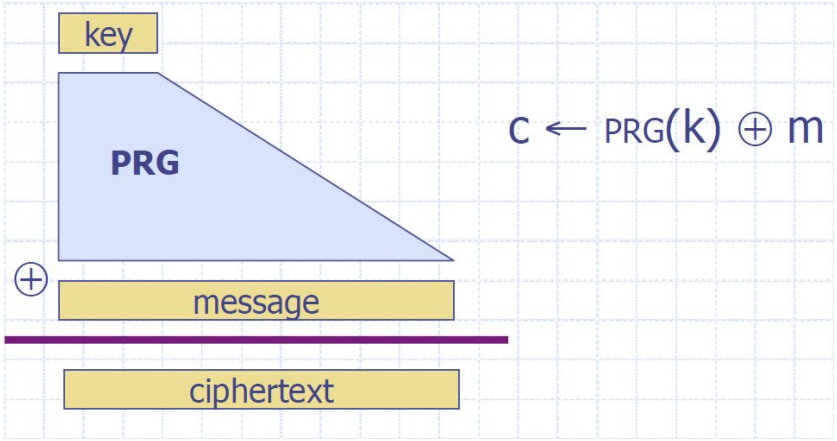


One-Time Pad

- One-time pad (OTP)
  - c = Encryption(m) = m XOR key
  - m = Decryption(c) = c XOR key
  - Key is random string at least as long as the plaintext
- Provides “perfect” secrecy in principle
- Practical disadvantages:
  - Requires truly uniform random one-time pad values
  - Keys must not be used more than once
  - Key length depends on the message length
  - Keys must be securely exchanged

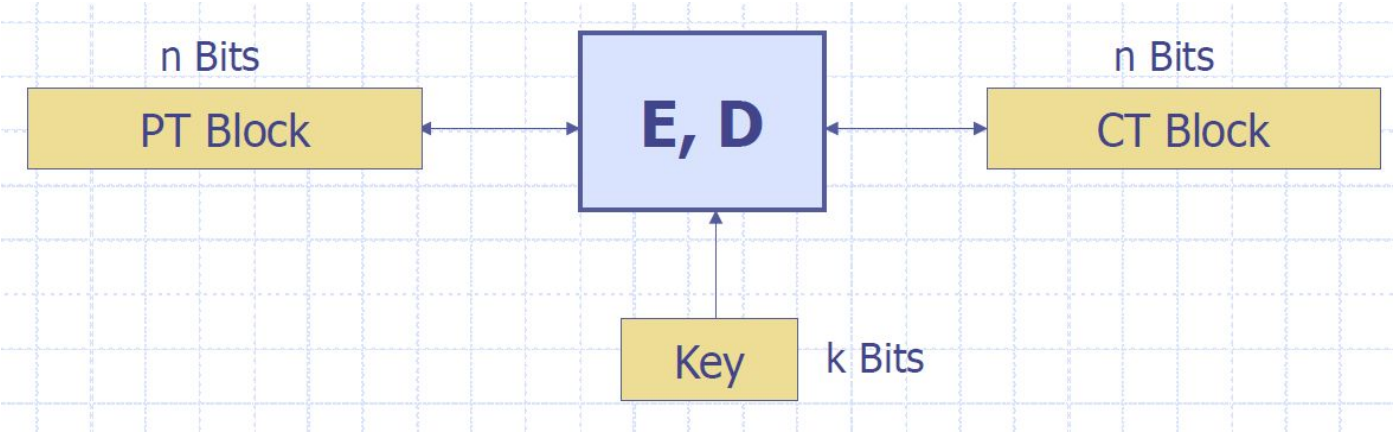
Stream Ciphers

- Making OTP practical (and less secure)
- Require Pseudo Random Generators
- Examples: RC4 (used in HTTPS and WEP), CSS (used for DVD encryption), E0 (used in Bluetooth), A5/1,2 (used in GSM encryption), Salsa20/ChaCha, …



Block Ciphers

- Encrypt blocks of data of fixed size
- Modes of operation handle variable length data

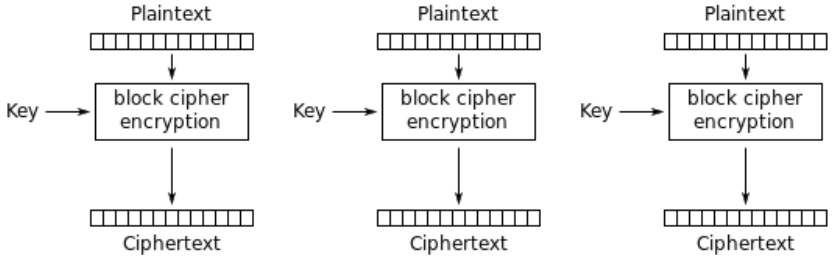


Examples of Block Ciphers

- Data Encryption Standard (DES):
  - Block size 64 bits
  - Key size 56 bits
  - Deprecated
- Advanced Encryption Standard (AES):
  - Block size 128 bits
  - Key size 128/192/256 bits
  - Hardware support in Intel and AMD processors

Modes of Operation - Electronic Code Book

- Electronic Code Book (ECB)
- Example of a *bad* mode of operation (insecure, obsolete): same plaintext blocks encrypt to same ciphertext blocks



Electronic Codebook (ECB) mode encryption

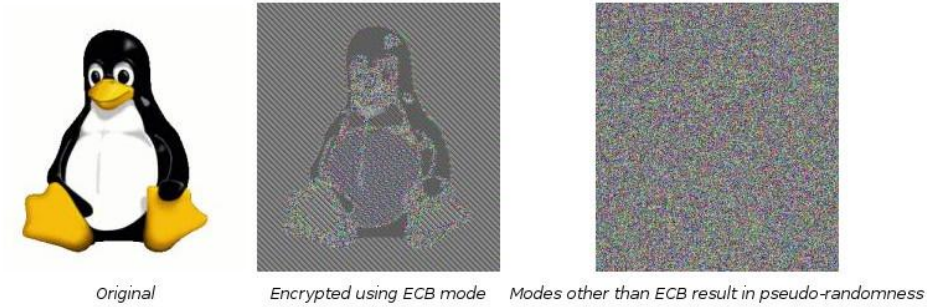
Original

ECB-encrypted image

User:Lunkwill of en.wikipedia

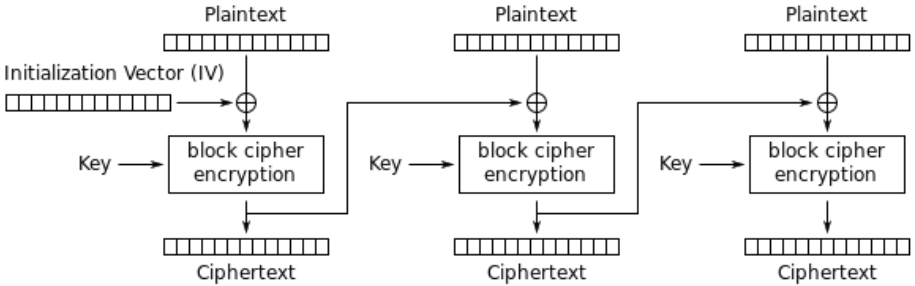
Modes of Operation - Cipher Block Chaining

- Cipher Block Chaining (CBC)
- A good, secure mode of operation (when used correctly)



Original *Encrypted using ECB mode* *Modes other than ECB result in pseudo-randomness*

User:Lunkwill of en.wikipedia



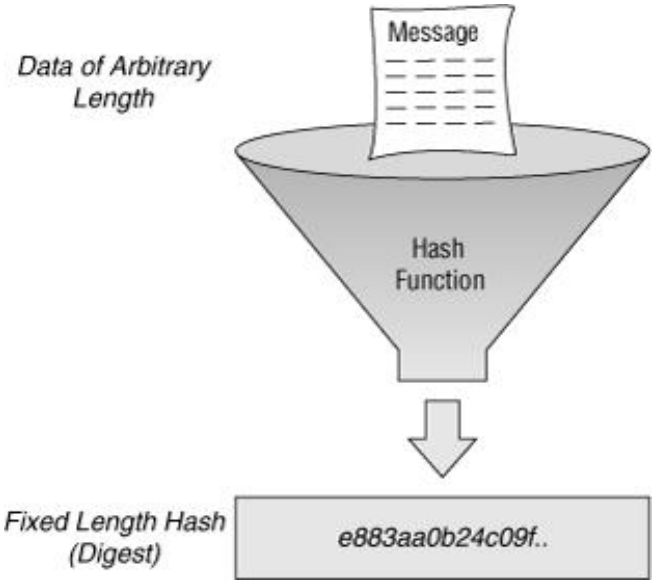
Cipher Block Chaining (CBC) mode encryption

Outline

- Introduction
- Symmetric crypto
- **Cryptographic hash functions**
- Data Integrity
- Authenticated encryption
- Public Key crypto
- Key distribution
- Public Key Infrastructure
- TLS
- HTTPS

Cryptographic hash functions

- Map bit-strings of any length to a fixed-length output in a deterministic way
- Desirable properties of hash functions (informal definitions):
  - **One-way:** given y it is infeasible to find any x such that y = h(x)
  - **Collision-resistance:** infeasible to find x and x' such that h(x)=h(x')
  - **Pseudo-randomness:** indistinguishable from a random oracle



[credit: Cisco]

SpeakUp #1 - Birthday Paradox

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

If you chose randomly 23 people, what is the probability of at least two having the same month and day in their birthday?

A. 10%
B. 25%
C. 50%
D. 95%

Dangers of Hash Functions

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- Why is a collision dangerous:
  - Seller generates two documents, with lower and higher price of the product, with the same hash. Buyer digitally signs the lower price document, but since both documents have the same hash, seller can now claim the the buyer signed the higher price document
  - Developer creates good and bad software patch, gets code-review and sign-off on good one, but slips the malicious one into the Git repository to create a secret software back-door
- Birthday paradox:
  - What is the probability that among 23 randomly chosen people, at least two of them have the same birthday? Answer: 50%
  - With just 70 people the probability is 99.9%
  - Relevant to hash functions - the expected number of N-bit hashes that can be generated before getting a collision is not 2^N, but rather only 2^(N/2)

Examples of Usage

- Password storage
- Files/Messages integrity verification
- Proof-of-work
- Key derivation
- Commitments
- Blockchains
- ...

(In)secure Hash Functions

- Examples: SHA-0, MD5, SHA-1, SHA-2, SHA-3, BLAKE, BLAKE2

Lifetimes of popular cryptographic hashes (the rainbow chart)

Function	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Snefru																												
MD2 (128-bit)[1]																												
MD4																												
MD5															[2]													
RIPEMD															[2]													
HAVAL-128[1]															[2]													
SHA-0																												
SHA-1																												[3]
RIPEMD-160																												
SHA-2 family																		[4]										
SHA-3 (Keccak)																												

Key Didn't exist/not public | Under peer review | Considered strong | Minor weakness | Weakened | Broken | Collision found

[1] Note that 128-bit hashes are at best 2^64 complexity to break; using a 128-bit hash is irresponsible based on sheer digest length.

[2] What happened in 2004? Xiaoyun Wang and Dengguo Feng and Xuejia Lai and Hongbo Yu happened.

[3] Google spent 6500 CPU years and 110 GPU years to convince everyone we need to stop using SHA-1 for security critical applications. Also because it was cool.

[4] In 2007, the NIST launched the SHA-3 competition because "Although there is no specific reason to believe that a practical attack on any of the SHA-2 family of hash functions is imminent, a successful collision attack on an algorithm in the SHA-2 family could have catastrophic effects for digital signatures." One year later the first strength reduction was published.

http://valerieaurora.org

(In)secure Hash Functions

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Reactions to stages in the life cycle of cryptographic hash functions			
Stage	**Expert reaction**	**Programmer reaction**	**Non-expert ("slashdotter") reaction**
Initial proposal	Skepticism, don't recommend use in practice	Wait to hear from the experts before adding to your crypto library	SHA-what?
Peer review	Moderate effort to find holes and garner an easy publication	Used by a particularly adventurous developers for specific purposes	Name-drop the hash at cocktail parties to impress other geeks
General acceptance	Top-level researchers begin serious work on finding a weakness (and international fame)	Even Microsoft is using the hash function now	Flame anyone who suggests the function may be broken in our lifetime
Minor weakness discovered	Massive downloads of turgid pre-prints from arXiv, calls for new hash functions	Start reviewing other hash functions for replacement	Long semi-mathematical posts comparing the complexity of the attack to the number of protons in the universe
Serious weakness discovered	Tension-filled CRYPTO rump sessions! A full break is considered inevitable	Migrate to new hash functions immediately, where necessary	Point out that no actual collisions have been found
First collision found	Uncork the champagne! Interest in the details of the construction, but no surprise	Gather around a co-worker's computer, comparing the colliding inputs and running the hash function on them	Explain why a simple collision attack is still useless, it's really the second pre-image attack that counts
Meaningful collisions generated on home computer	How adorable! I'm busy trying to break this new hash function, though	Send each other colliding X.509 certificates as pranks	Claim that you always knew it would be broken
Collisions generated by hand	Memorize as fun party trick for next faculty mixer	Boggle	Try to remember how to do long division by hand
Assumed to be weak but no one bothers to break	No one is getting a publication out of breaking this	What's this crypto library function for?	Update Pokemon Wikipedia pages

http://valerieaurora.org

Outline

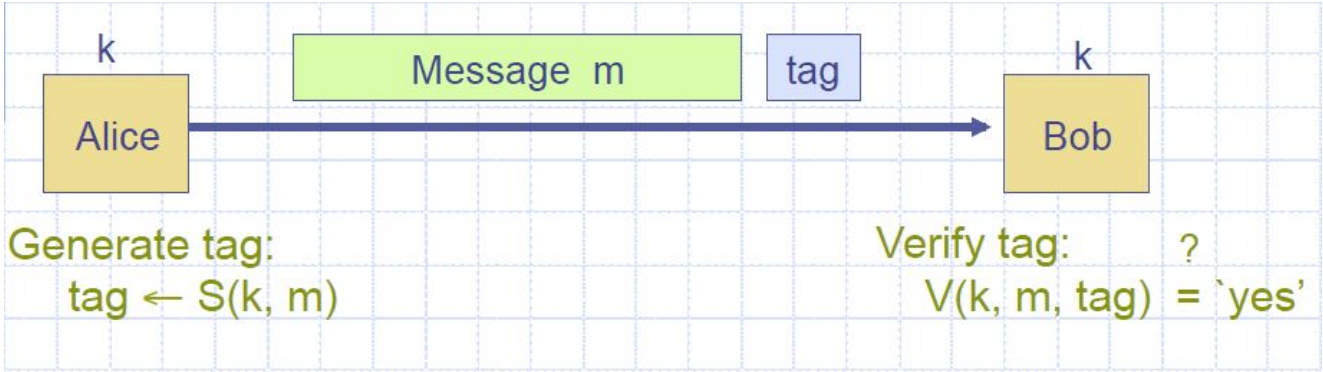
- Introduction
- Symmetric crypto
- Cryptographic hash functions
- **Data Integrity**
- Authenticated encryption
- Public Key crypto
- Key distribution
- Public Key Infrastructure
- TLS
- HTTPS

Data integrity

- Goal: provide integrity, not confidentiality
- Integrity - ensure that data is not modified by unauthorized parties
- Symmetric crypto doesn’t provide integrity by itself:
  - Example: flipping a bit in OTP cipher text results in flipping the same bit in the plaintext after decryption
  - Examples for block ciphers are a bit more complicated, but neither block ciphers provide integrity

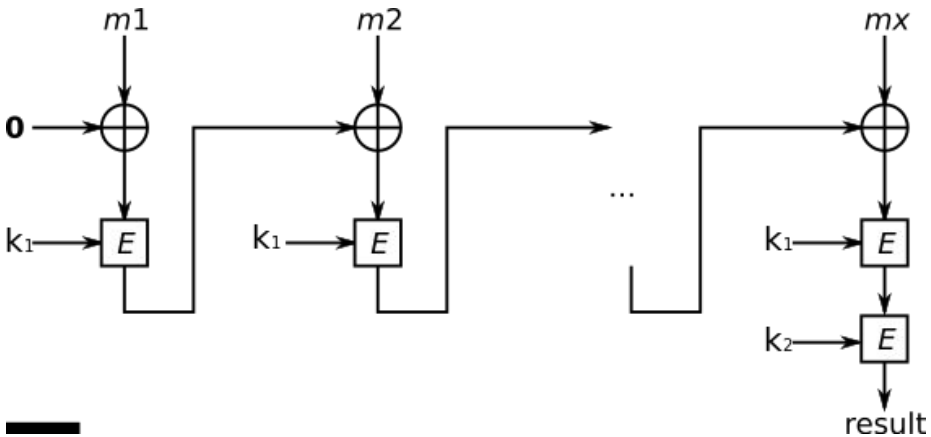
Message Authentication Codes

- Message Authentication Codes (MACs):
- Alice sends a message to Bob, such that Bob is able to verify that the message was sent by Alice and that the message is not modified in transit
- Alice and Bob have a shared secret key



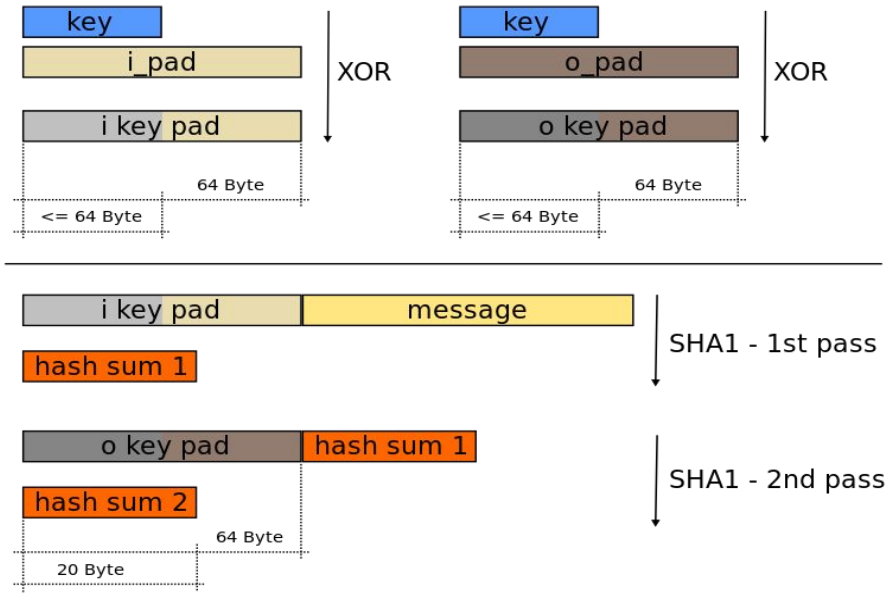
ECBC-MAC

- Encrypt-last-block CBC-MAC
- Last block of CBC encryption of the message + the additional encryption step
- Used in banking systems
- Standards: ANSI x9.9, x9.19, FIPS 186-3
- Often based on the AES algorithm



Hash Message Authentication Code

- Hash Message Authentication Code (HMAC)
- Based on a cryptographic hash function
- Examples: HMAC_MD5, HMAC_SHA1, …
- Used in TLS, IPsec, …
- Standards: FIPS PUB 198



Outline

- Introduction
- Symmetric crypto
- Cryptographic hash functions
- Data Integrity
- **Authenticated encryption**
- Public Key crypto
- Key distribution
- Public Key Infrastructure
- TLS
- HTTPS

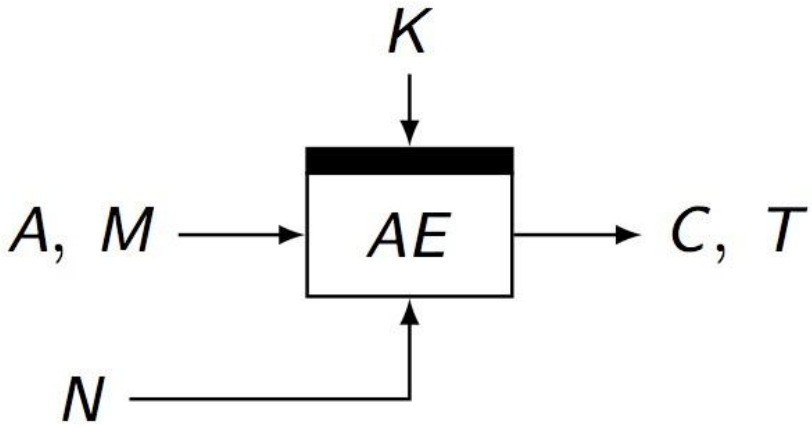
Authenticated encryption

- So far, we've seen tools to obtain confidentiality and tools to obtain integrity
- Goal: ensure confidentiality and integrity together → Authenticated Encryption
  - Less error-prone primitive to use as "black box"
- AE abstraction introduced in 2000
- Two main ways to construct AE schemes:
  - Combine MAC and unauthenticated encryption schemes:
    - MAC-then-Encrypt (MtE) - SSL
    - Encrypt-and-MAC (E&M) - SSH
    - Encrypt-then-MAC (EtM) - IPSec
      - Strongest in formal analysis for arbitrary algorithm composition
  - Create fresh AE schemes "from scratch" for this purpose
    - CAESAR cipher competition currently ongoing

Authenticated Encryption [w/ Additional Data]

Parameters:

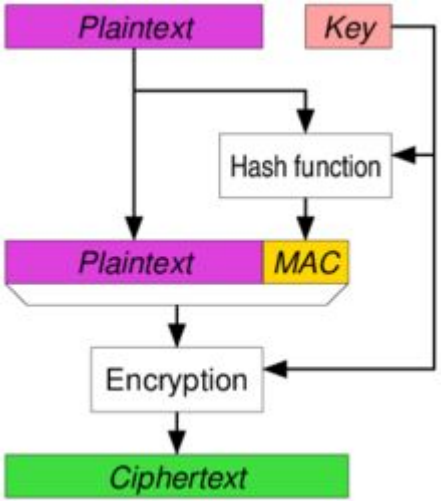
- K: key
- A, M: additional data and message
- N: nonce
- C, T: ciphertext and tag



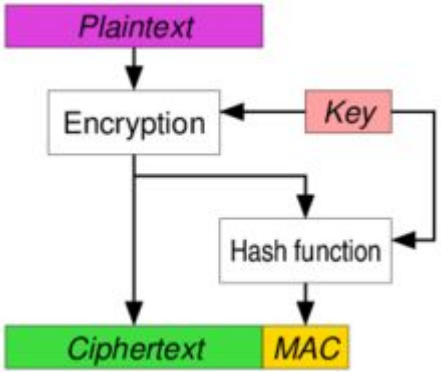
Properties:

- Message M is **both** authenticated (by tag T)
  and encrypted (to ciphertext C)
- Additional data A is **only** authenticated (by tag T)
  but not encrypted

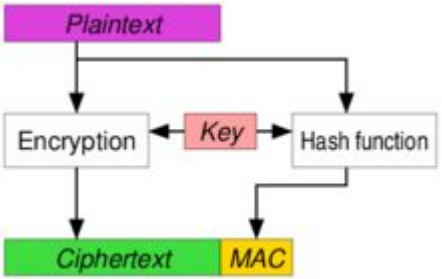
Three Approaches - Detail



MtE

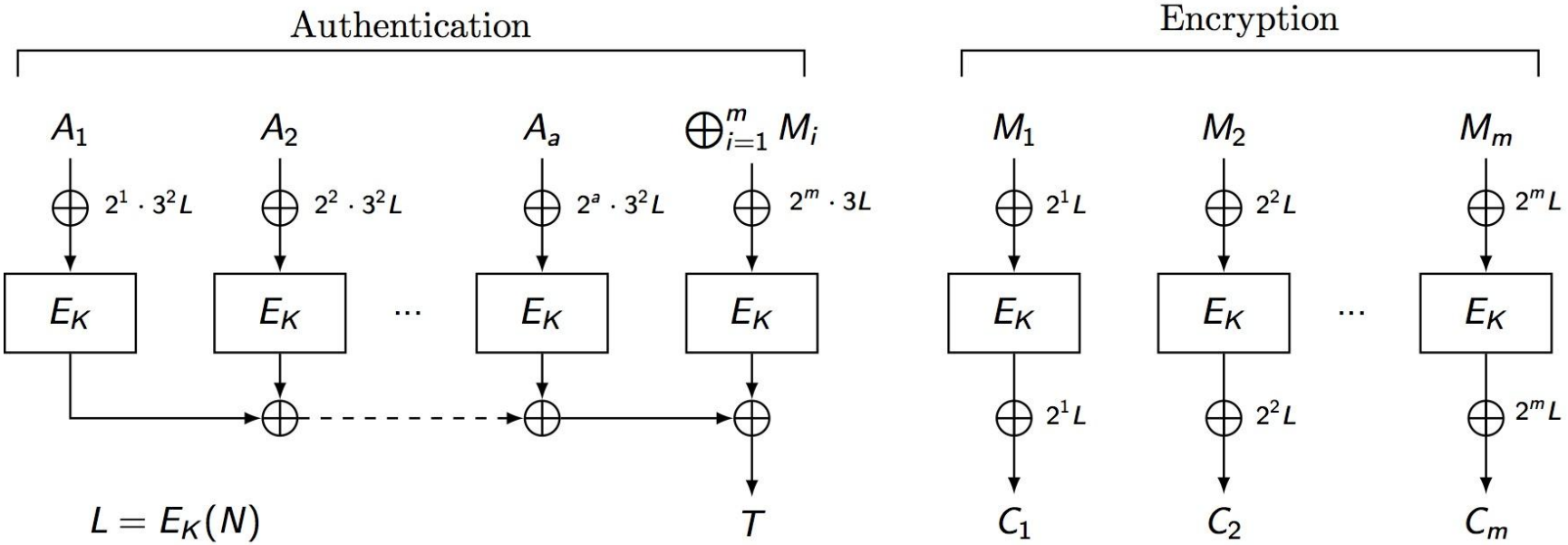


EtM



E&M

Offset Codebook (OCB) Mode



- AEAD mode
- Based on block cipher E_k (usually AES)
- Offset masks ($2^i * 3^j * L$) for domain separation between data blocks

Outline

- Introduction
- Symmetric crypto
- Cryptographic hash functions
- Data Integrity
- Authenticated encryption
- **Public Key crypto**
- Key distribution
- Public Key Infrastructure
- TLS
- HTTPS

Public Key Crypto

- All the crypto we've seen until now assumes that Alice and Bob have a shared secret key
- What happens if they don't?
  - They can exchange the key over a secure channel, but the problem is how to obtain this channel (chicken-and-egg problem)
  - We can assume that Alice and Bob have only an insecure channel between them, so we can't just send the key over this channel, because Eve could obtain it
- Interesting rumor: the red phone connecting Moscow and Washington was encrypted with OTP, and the keys were shared by diplomats who flew from one country to the other carrying briefcases full of print-outs of shared keys
  - Not quite practical for everyone wanting to order a plushie from Amazon.com

First Idea of Asymmetric Crypto

- 1975. Diffie and Hellman in “New directions in cryptography” describe the idea of asymmetric (public key) cryptography:

We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals.

In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

Interesting Facts

- DH key exchange was published in 1976
- RSA scheme was published in 1977

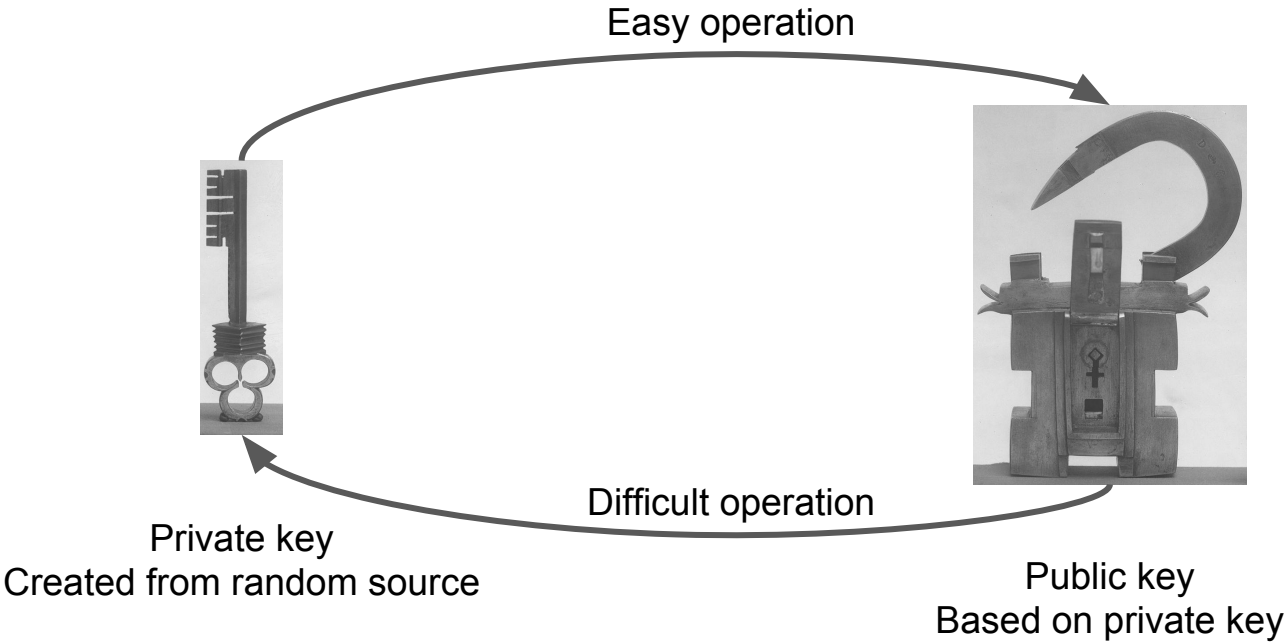
Rewind...

- 1970. James Ellis at UK GCHQ realises that the idea of public key crypto is possible, in a secret report “The possibility of Non-secret Encryption”
- 1973. Clifford Cocks at UK GCHQ discovers a workable mathematical formula for non-secret encryption in a secret report “A note on Non-Secret Encryption”. His formula was a special case of the RSA algorithm
- 1974. Malcolm Williamson at UK GCHQ describes a key exchange method in a secret report “Non-Secret Encryption Using a Finite Field”. His method was similar to the one discovered by Diffie and Hellman

Primitives

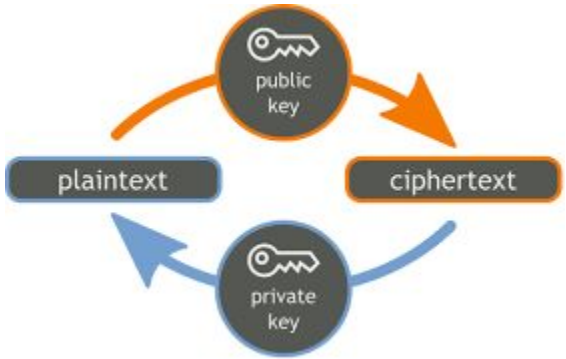
- Public and private key
  - Two keys (numbers), *public* is distributed widely, *private* is kept secret
  - Easy: f(private) -> public
  - Hard: f(public) -> private
- Encryption and Decryption
  - Encrypt with public key, decrypt with private key
  - Hard to decrypt without the private key
- Digital signatures
  - Sign with the private key, verify the signature using the public key
  - Hard to create a signature if only public key is known
- Interactive key exchange
  - Create a shared *private* key over an insecure communication channel

Private and Public Keys



Public Key Encryption

- Cryptosystem with two keys, public and private key
- Public key is used by a sender to encrypt a message, while the secret key is used by a receiver to decrypt the ciphertext and recover the message
- Even an adversary which knows the public key can not decrypt the ciphertext, it can only be decrypted with the private key



Comparison with Symmetric Crypto

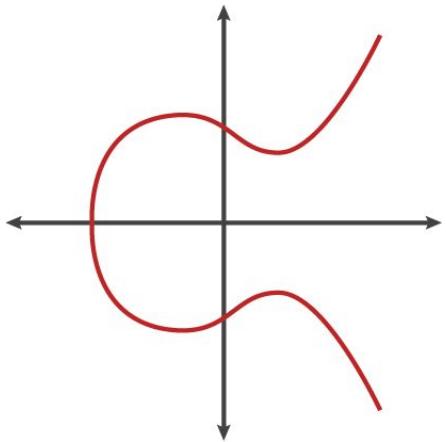
- Advantages relative to symmetric crypto
  - Communicating parties don't need to share a secret key before their communication
  - If we have more communicating parties, we don't need shared secret keys between each pair: each party has its public key which others can use to communicate with this party
- Disadvantages relative to symmetric crypto:
  - Performance: symmetric crypto is much more computationally efficient byte-for-byte
  - Storage: keys and message overheads usually higher for public-key crypto
  - Fragility: public-key crypto relies on algebraic operations being (and staying) "hard"
    - Some of these assumptions become easy if/when quantum computers emerge

Rivest, Shamir, Adleman

- Public key encryption scheme - RSA:
- 1977. Ronald Rivest, Adi Shamir, Len Adleman
  - Pick large primes *p* and *q* and compute *n = p*q* and *phi_n = lcm(m-1, n-1)*
  - Pick secret key *sk* coprime with *m-1* and *n-1*, and compute public key *pk = sk^-1 mod(phi_n)*
  - Keep *sk* secret and publish *pk*
  - Encryption(*m*) = *m^pk mod(n)*
  - Decryption(*c*) = *c^sk mod(n)*
- Relies on hardness of the factorization problem - given *n*, where *n = p*q* and *p* and *q* are large prime numbers, it is very hard to find *p* and *q*
- Note: the ‘plain’ RSA described above is not secure; in practice it is used with some modifications

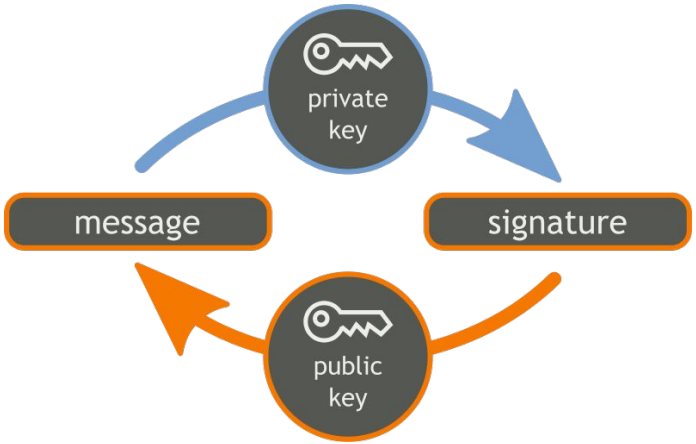
Elliptic Curve Cryptography

- Elliptic curve cryptography (ECC) is based on the algebraic structure of elliptic curves over finite fields.
  - General elliptic curve: $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$
  - Montgomery curve: $y^2 = x^3 + ax^2 + x$
- Smaller keys for equivalent security
  - 256-bit for ECC comparable to 2048-bit RSA
- Faster operations than RSA for large keys
- Some curves (esp. over GF(2^n)) turned out to be weak (e.g., RFC2409, group 3&4)
- Popular secure curves (over GF(p), p prime):
  - Curve25519
  - Curve448



Signatures

- Digital signatures
- Public key counterpart of MACs (with important differences)
- Provide a way for a signer *S* with a public and private key pair (*pk, sk*) to sign a message in such a way that any other party who knows *pk* can verify that *S* is the author of the message and that the message was not modified



Signatures - Motivation

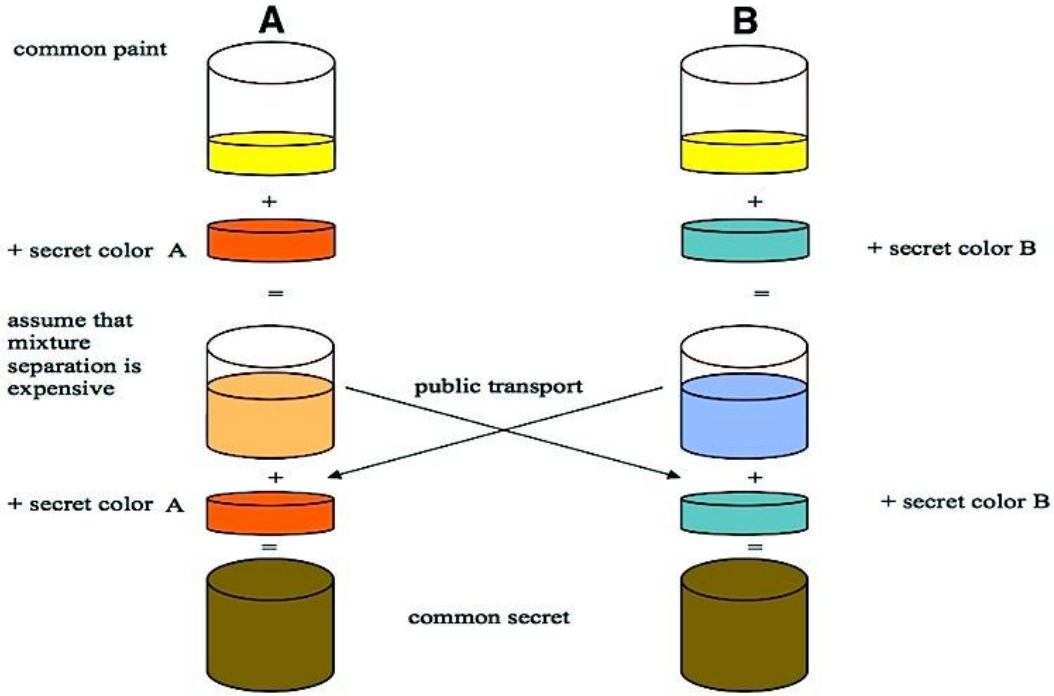
- Software/Update distribution
  - Company releases an update for its product. Users can download the update, but before installing it on their machines, the users should be able to verify that the update indeed came from the company and that it is not modified
- Authenticated email
  - Provide a way for Alice to verify that the email she received from Bob is indeed from Bob and not from someone else
  - Symmetric crypto already solves this but users must share a secret key
- Certificates

Signatures Example - RSA

- Pick large primes *p* and *q* and compute *n = p*q* and *phi_n = lcm(m-1, n-1)*
- Pick secret key *sk* coprime with *m-1* and *n-1*, and compute public key *pk = sk^-1 mod(phi_n)*
- Keep *sk* secret and publish *pk*
- Sign(*m*) = *m^sk mod(n)*
- Verify(*c*): *c^pk mod(n) ?= m*

Interactive Key Exchange

- Diffie-Hellman Key Exchange



Credit: A.J. Han Vinck, Introduction to public key cryptography

Diffie-Hellman Key Exchange

- Steps
  - Alice and Bob agree on a finite algebraic group *G* with generator *g*
  - Alice picks a random *a* and sends *g^a* to Bob
  - Bob picks a random *b* and sends *g^b* to Alice
  - Alice computes *s = (g^b)^a*
  - Bob computes *s = (g^a)^b*
- Security of DH relies on hardness of the Discrete Logarithm Problem
- DLP is not hard in all groups: must choose appropriately
- Application example: Handshake protocol in TLS

SpeakUp #2

What should you use to store users' passwords in a database?

A. Hash
B. Encryption with a key
C. Signature
D. None of the above

Outline

- Introduction
- Symmetric crypto
- Cryptographic hash functions
- Data Integrity
- Authenticated encryption
- Public Key crypto
- **Key distribution**
- Public Key Infrastructure
- TLS
- HTTPS

Key Distribution

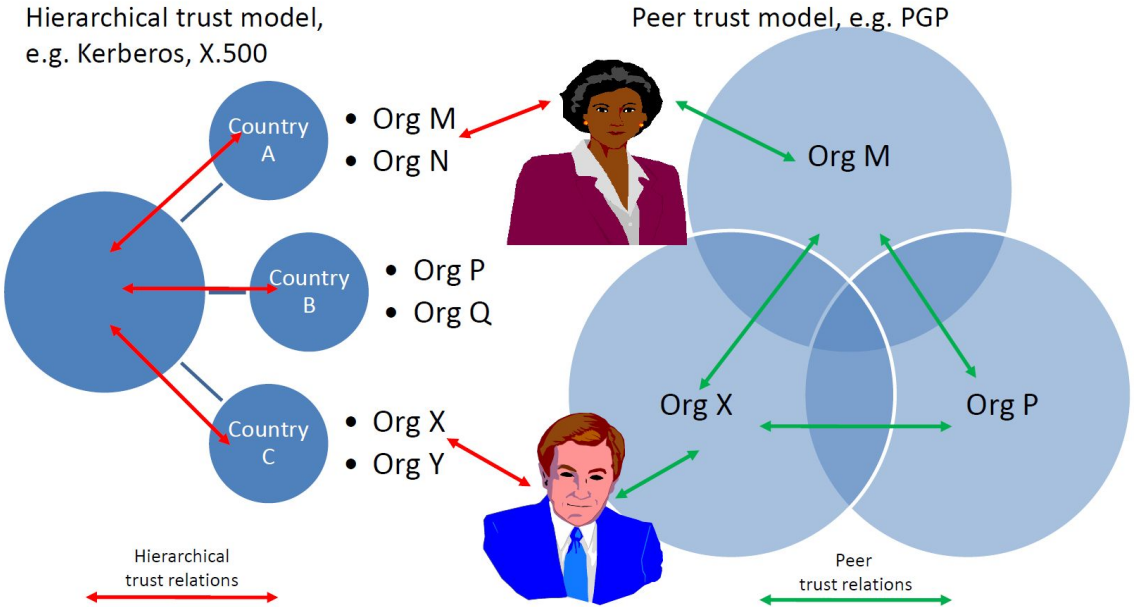
- For both symmetric and asymmetric crypto we have to distribute keys
- Symmetric cryptosystems require the exchange of secret keys
  - Need for a secret/confidential channel
- Asymmetric cryptosystems require the exchange of public keys
  - Need for a trusted/integrity protected channel
- Authorities trusted to provide secret / trustworthy keys:
  - Key Distribution Centers (KDC)
  - Certification Authorities (CA)

How to Distribute Trust

- KDCs provide shared secret keys through confidential tickets:
  - Ticket from KDC
  - To Alice
  - For secret key with Bob
  - E_a(Bob, K_ab, key attributes)
  - To get a shared secret key Alice and Bob each need to have a shared secret key with KDC
- CAs provide trusted public keys through signed certificates:
  - Certificate from CA
  - To Alice
  - For public key of Bob
  - S_k(Bob, P_b, key attributes)
  - To get each other's public key, Alice and Bob each need to have the public key of trusted CA

Using Hierarchy of Trust

- One KDC/CA is not enough to serve all users
- KDCs/CAs are organized into hierarchies or peer networks



Outline

- Introduction
- Symmetric crypto
- Cryptographic hash functions
- Data Integrity
- Authenticated encryption
- Public Key crypto
- Key distribution
- **Public Key Infrastructure**
- TLS
- HTTPS

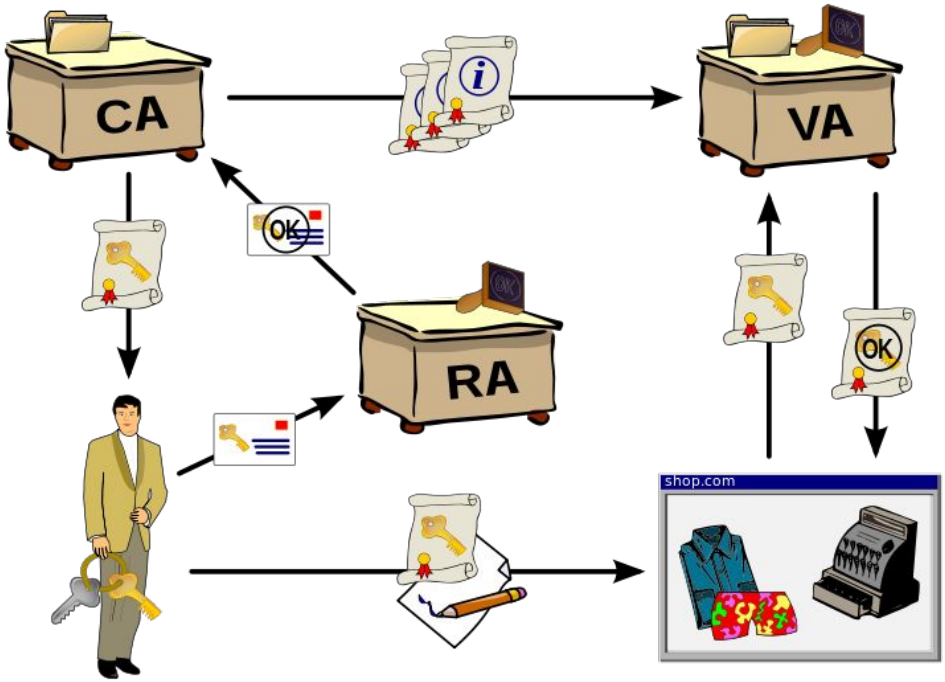
Public Key Infrastructure

PKI is a set of of roles, policies, procedures to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption

PKI binds public keys with their owners through a process that includes:

- Digital certificate: electronic document proving ownership of a public key
- Certificate Authority (CA): stores, issues and signs the digital certificates
- Registration Authority (RA): verifies the identity of entities requesting their digital certificates to be stored at the CA
- Certificate Policy (CP): defines purpose(s) for which certificate is to be used

Elements of PKI



Methods Used for PKI

- Certificate Authorities (CAs):
  - CAs sign and publish user's public key with their own private key
  - Trust in the user key relies on the trust in the CA's key
  - CA = Trusted Third Party
- Web of Trust (WOT):
  - PGP, GnuPG
  - Self-signed certificates
  - Third parties validate certificates
- Simple Public-Key Infrastructure (SPKI):
  - experimental, tries to overcome the complexities of CAs and WOT

Attacks

- Huge problem when CA gets compromised (Comodo, DigiNotar)

theguardian

UK world sport football opinion culture business lifestyle fashion environment tech travel

browse all sections

home › tech

Hacking

DigiNotar SSL certificate hack amounts to cyberwar, says expert

Dutch government revokes certificates used for all its secure online transactions, while CIA, Google, Microsoft and others affected by hack called 'worse than Stuxnet'

This article is 5 years old

0 10

Charles Arthur and agencies

@charlesarthur

Monday 5 September 2011 18.14 BST

Aanpak DigiNotar-problematiek

Nieuws

Belastingen

Landbouw, natuur en voedsel

Bouwen, wonen en leefomgeving

Milieu, ruimte en water

Uitgelicht

Voor publieksvragen over DigiNotar bel 0800 1351

The Dutch government has revoked all trust in digital certificates issued by DigiNotar

Most popular in US

Minnesota Vikings' new glass-plated stadium becomes 'death trap' for birds

Uber CEO Travis Kalanick caught on video arguing with driver about fares

Increased risk of 11 types of cancer linked to being overweight, researchers warn

Detained by US immigration: 'In that

Outline

- Introduction
- Symmetric crypto
- Cryptographic hash functions
- Data Integrity
- Authenticated encryption
- Public Key crypto
- Key distribution
- Public Key Infrastructure
- **TLS**
- HTTPS

TLS - Definition

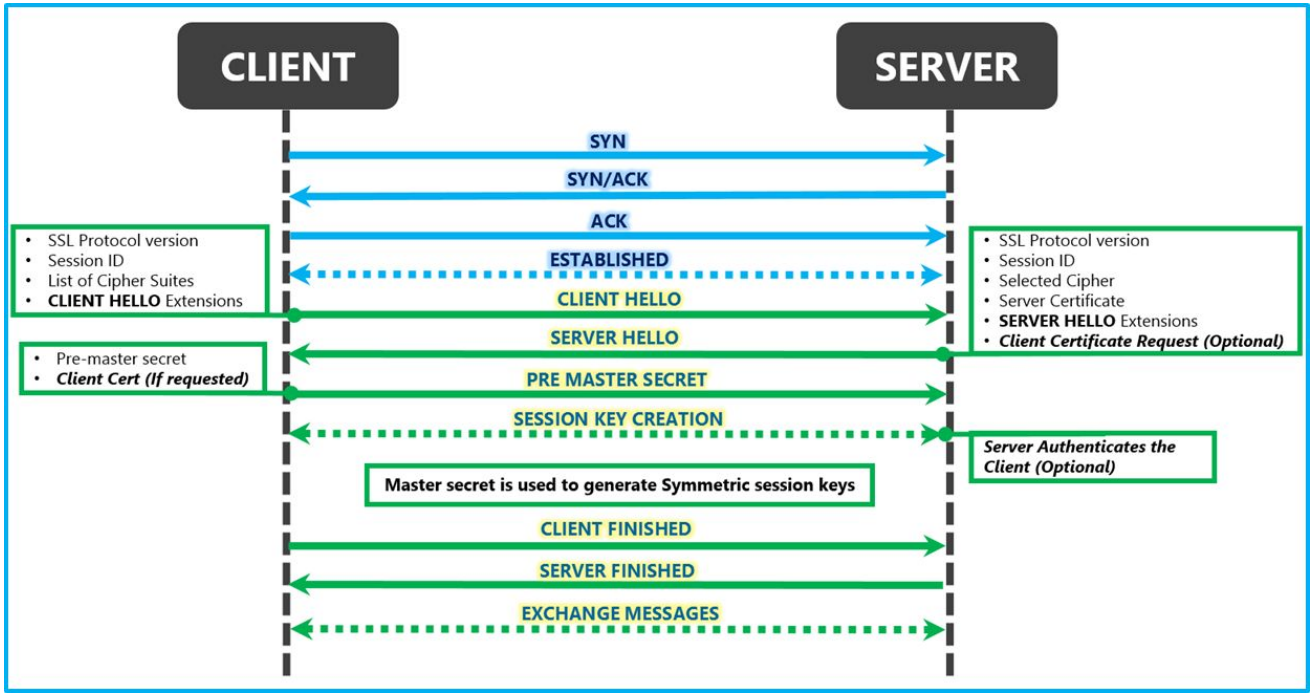
- Transport Layer Security / Secure Sockets Layer
- “TLS protocol aims primarily to provide privacy and data integrity between two communicating computer applications”
- Widely used: Web browsing, email, VoIP, instant messaging, etc.
- History overview:
  - SSL 1.0, 2.0, 3.0 - developed at Netscape in 95/96 (deprecated)
  - TLS 1.0 - defined in 1999 in RFC 2246
  - TLS 1.1 - defined in 2006 in RFC 4346
  - TLS 1.2 - defined in 2008 in RFC 5246
  - TLS 1.3 - working draft since July 2016

Usage

- TLS offers the following properties for TLS secured connections:
  - Private/Secure connection: data transmitted between the communicating parties is encrypted with one of the supported symmetric crypto algorithms
  - Authentication of the communicating parties by use of the public key crypto algorithms, authentication is optional, but it is usually required for at least one of the parties
  - Integrity: transmitted messages include MAC and can be verified for integrity
  - Forward secrecy: attacker who later breaks/steals private key can’t decrypt old conversations
- Methods for key exchange, encryption algorithms and MACs can be chosen by various configurable parameters
- Not all parameter choices will provide all the security properties

Example

- Client and server run the TLS handshake protocol to negotiate a connection



Some Historical TLS Vulnerabilities

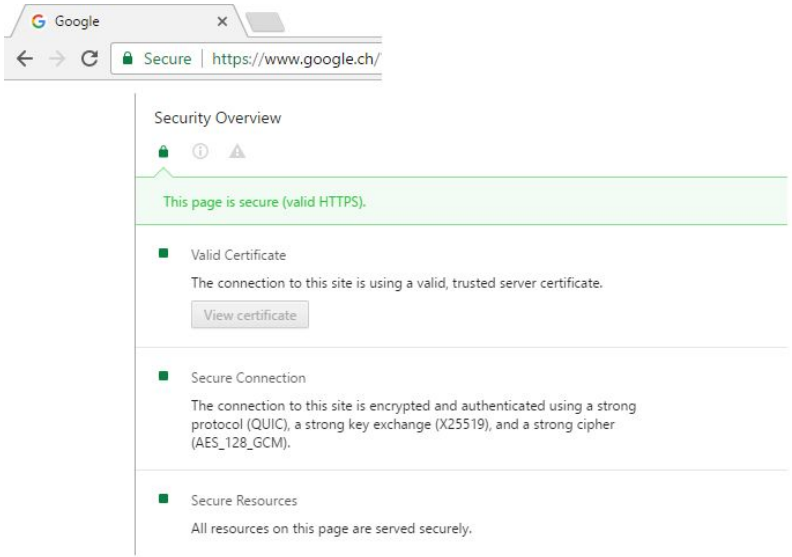
- Downgrade attack:
  - Trick the server into using an insecure version of TLS
  - Negotiate a connection with weak keys (FREAK, 2014.)
  - Force the server to do downgrade to weak DH groups (Logjam, 2015.)
- CRIME, BREACH attacks - exploiting the vulnerability when data compression is used with TLS
- POODLE, 2014. - SSL 3.0 vulnerability
- Bugs in Implementations:
  - 2014. Heartbleed in OpenSSL
  - 2017. Cloudbleed - a bug in Cloudflare HTML parser allowed people to read data in memory of programs running on the servers

Outline

- Introduction
- Symmetric crypto
- Cryptographic hash functions
- Data Integrity
- Authenticated encryption
- Public Key crypto
- Key distribution
- Public Key Infrastructure
- TLS
- **HTTPS**

HTTPS

- HTTP over TLS, HTTP over SSL, HTTP Secure
- Communication over the HTTP where connection is encrypted by TLS



Usage

- HTTPS provides the authentication (usually of the web server) and privacy and integrity of data exchanged between the user and the server
- Applications of HTTPS:
  - Internet banking, payments
  - E-commerce, e.g., Amazon.com
  - Filling forms with personal/sensitive information
  - Public Wi-Fi spots - anyone on the network can sniff packets
- Stats:
  - September 2016: 11.6% of Alexa top 1M websites use HTTPS as default

Attacks and Defenses 1/2

- Security of HTTPS relies on the security of TLS
- Web browsers have a pre-installed list of trusted Certificate Authorities
- CAs are a major concern and a potential point of failure to MiTM attacks:
  - e.g. French CA “Tresor public” issued fake certificates impersonating Google to the French government, in order to spy on government employees via MiTM attack
  - Google Certificate Transparency project tries to reveal problems quickly
- For a website to be secure, it must be completely hosted over HTTPS
  - This is a problem for many websites which earn from ads, because some of the ad-networks don’t support HTTPS
  - EPFL is still not 100% HTTPS-compliant

Attacks and Defenses 2/2

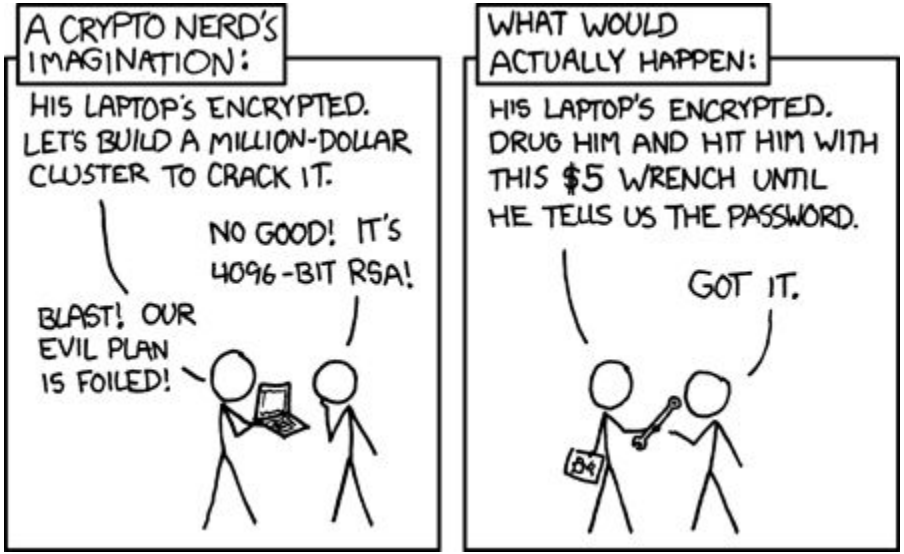
- SSL-stripping: MiTM attack which downgrades victim's web connection to the server from HTTPS to HTTP
  - Presented by Moxie Marlinspike at Black Hat 2009
  - User can see that the connection is insecure (over HTTP), but doesn’t know if the connection should be secure (over HTTPS)
  - Browser doesn’t warn about this, so the attack is quite effective
- HTTP Strict Transport Security (HSTS)
  - Policy mechanism used to protect against protocol downgrade attacks and cookie hijacking
  - Protects from SSL-stripping
  - HSTS tells the browser that the connection to the website should always be over HTTPS, so the browser can alert the user if SSL-stripping happened

SpeakUp #3

What is your current state of protection using encryption?

A. Full hard disk encryption
B. Home-directory encrypted
C. Encrypted and signed emails
D. None of the above - sharing is caring

Conclusion


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.