APNIC 44

Crypto 101

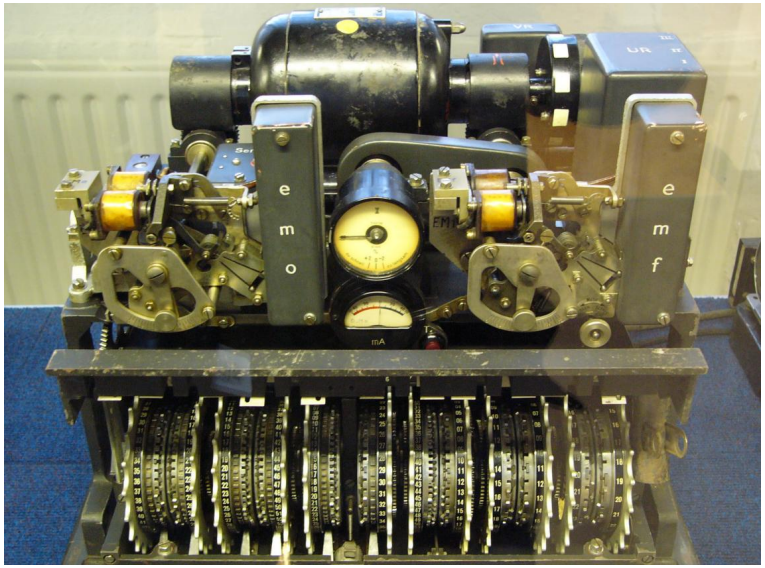
TAICHUNG, TAIWAN

7–14 September 2017

#apnic44

Cryptography

- Cryptography is everywhere



German Lorenz cipher machine

Cryptography

- Cryptography deals with creating documents that can be shared secretly over public communication channels
- Other terms closely associated
  - Cryptanalysis = code breaking
  - Cryptology
    - Kryptos (hidden or secret) and Logos (description) = secret speech / communication
    - combination of cryptography and cryptanalysis
- Cryptography is a function of plaintext and a cryptographic key

$$C = F(P,k)$$

Notation:

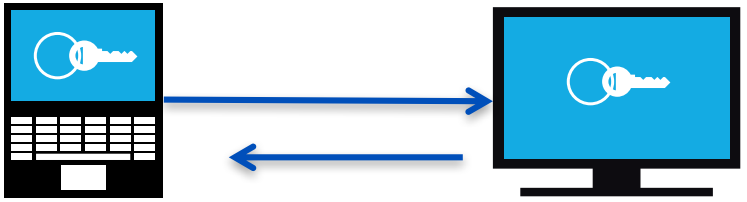
Plaintext (P)
Ciphertext (C)
Cryptographic Key (k)

Typical Scenario

- Alice wants to send a “secret” message to Bob
- What are the possible problems?
  - Data can be intercepted
- What are the ways to intercept this message?
- How to conceal the message?
  - Encryption

Crypto Core

- Secure key establishment

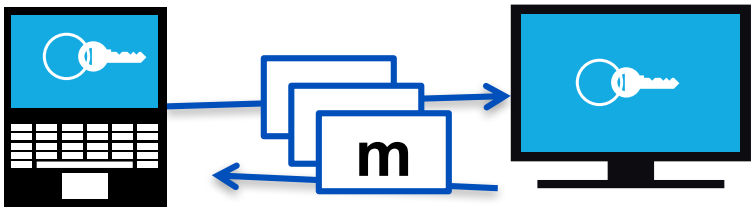


Alice has key (k)

Bob has key (k)

- Secure communication

Confidentiality and integrity



Alice has key (k)

Bob has key (k)

Source: Dan Boneh, Stanford

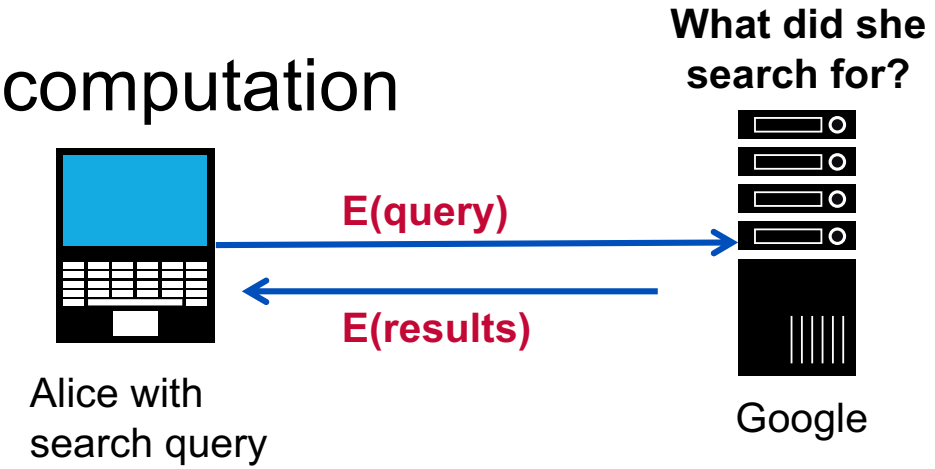
It can do much more

- Anonymous communication
- Anonymous digital cash
  - Spending a digital coin without anyone knowing my identity
  - Buy online anonymously?
  - Cryptocurrency / Bitcoin?
- Elections and private auctions
  - Finding the winner without actually knowing individual votes (privacy)

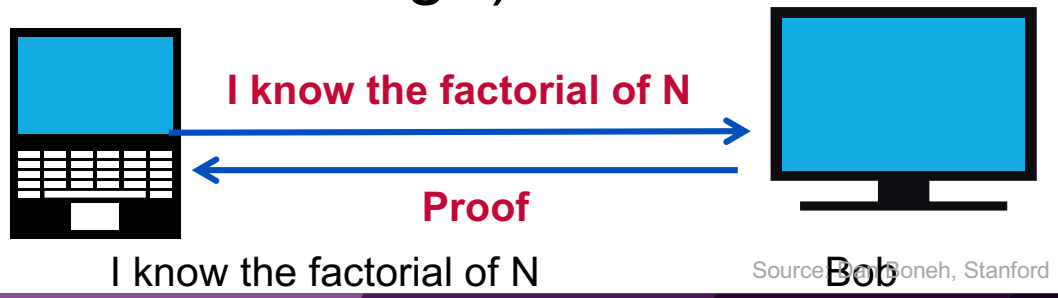
Source: Dan Boneh, Stanford

Crypto Magic

- Privately outsourcing computation



- Zero knowledge (proof of knowledge)



Source: Dan Boneh, Stanford

History: Ciphers

- Substitution cipher
  - involves replacing an alphabet with another character of the same alphabet set
  - Can be mono-alphabetic (single set for substitution) or poly-alphabetic system (multiple alphabetic sets)
- Example:
  - Caesar cipher, a mono-alphabetic system in which each character is replaced by the third character in succession
  - Vigenere cipher, a poly-alphabetic cipher that uses a 26x26 table of characters

How to Break a Substitution Cipher

UKBYBIPOUZBCUFEEBORUKBYBHOBBRFESPVKBWFOFERVNBCVBZPRUBOFERVNBCVBPCYYFVUFO

FEIKNWFRFIKJNUPWRFIPOUNVNIPUBRNCUKBEFWWFDNCHXCYBOHOPYXPUBNCUBOYNRVNIWN

CPOJIOFHOPZRVFZIXUBORJRUBZRBCHNCBBONCHRJZSFWNVRJRUBZRPCYZPUKBZPUNVPWPCYVF

ZIXUPUNFCPWRVNBCVBRPYYNUNFCPWWJUKBYBIPOUZBCUIPOUNVNIPUBRNCHOPYXPUBNCUB

OYNRVNIWNCPOJIOFHOPZRNCRVNBCUNENVVFZIXUNCHPCYVFZIXUPUNFCPWZPUKBZPUNVR

(1) Use frequency of the English letters

e = 12.7%
t = 9.1 %
a = 8.1%

(2) Use frequency of pairs of letters

he, in, an, th

In the example,
B appeared 36 times, **U** 33 times, and **P** 32 times
NC appeared 11 times, **PU** 10 times
UKB appeared 6 times

Source: Dan Boneh, Stanford

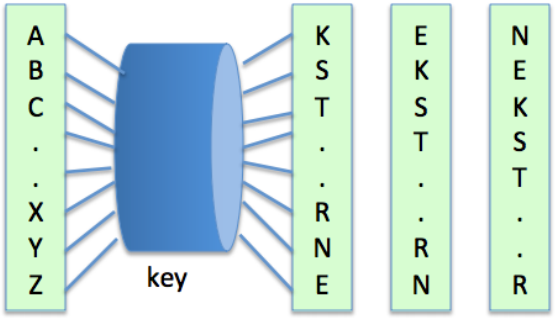
Transposition Cipher

- No letters are replaced, they are just rearranged.
- Rail Fence Cipher – another kind of transposition cipher in which the words are spelled out as if they were a rail fence.

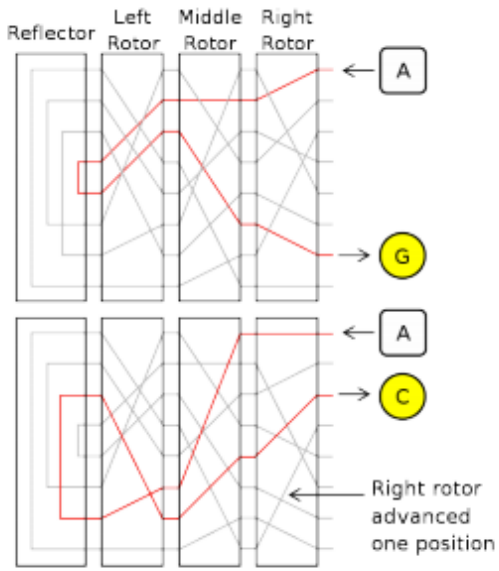
```
T...U...B...N...J...E...E...E...Y..
.H.Q.I.K.R.W.F.X.U.P.D.V.R.H.L.Z.D.G.
..E...C...O...O...M...O...T...A...O
```

History: Rotor Machines (1870-1943)

- Hebern machine – single rotor



- Enigma - 3-5 rotors



Source: Wikipedia (image)

Modern Crypto Algorithms

- specifies the mathematical transformation that is performed on data to encrypt/decrypt
- Crypto algorithm is NOT proprietary
- Analyzed by public community to show that there are no serious weaknesses
- Explicitly designed for encryption

Kerckhoff's Law (1883)

- The system must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.

- In other words, the security of the system must rest entirely on the secrecy of the key.

Properties of a Good Cryptosystem

- There should be no way short of enumerating all possible keys to find the key from any amount of ciphertext and plaintext, nor any way to produce plaintext from ciphertext without the key.
- Enumerating all possible keys must be infeasible.
- The ciphertext must be indistinguishable from true random values.

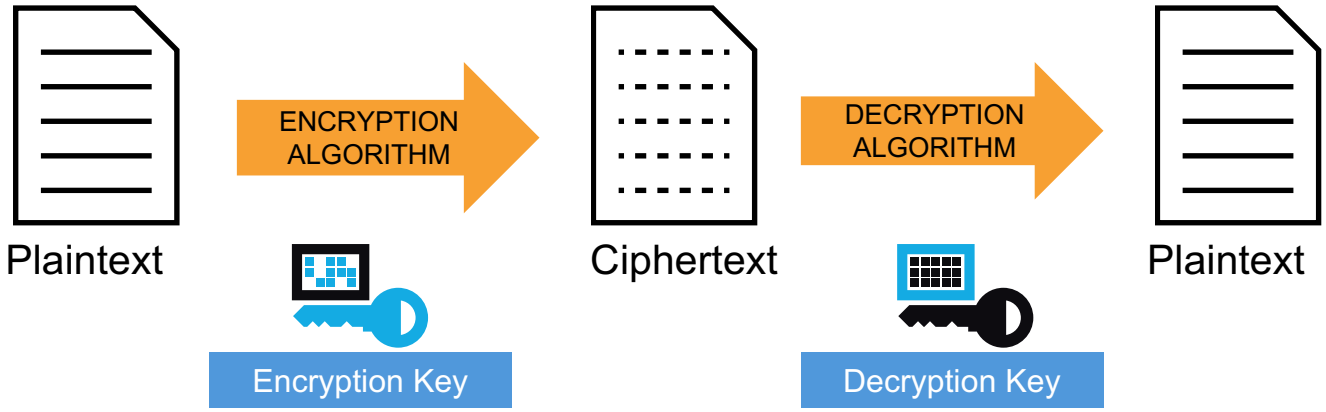
Encryption

- process of transforming plaintext to ciphertext using a cryptographic key
- Used all around us
  - In Application Layer – used in secure email, database sessions, and messaging
  - In session layer – using Secure Socket Layer (SSL) or Transport Layer Security (TLS)
  - In the Network Layer – using protocols such as IPsec

Encryption – Benefits

- Resistant to cryptographic attack
- They support variable and long key lengths and scalability
- They create an avalanche effect
- No export or import restrictions

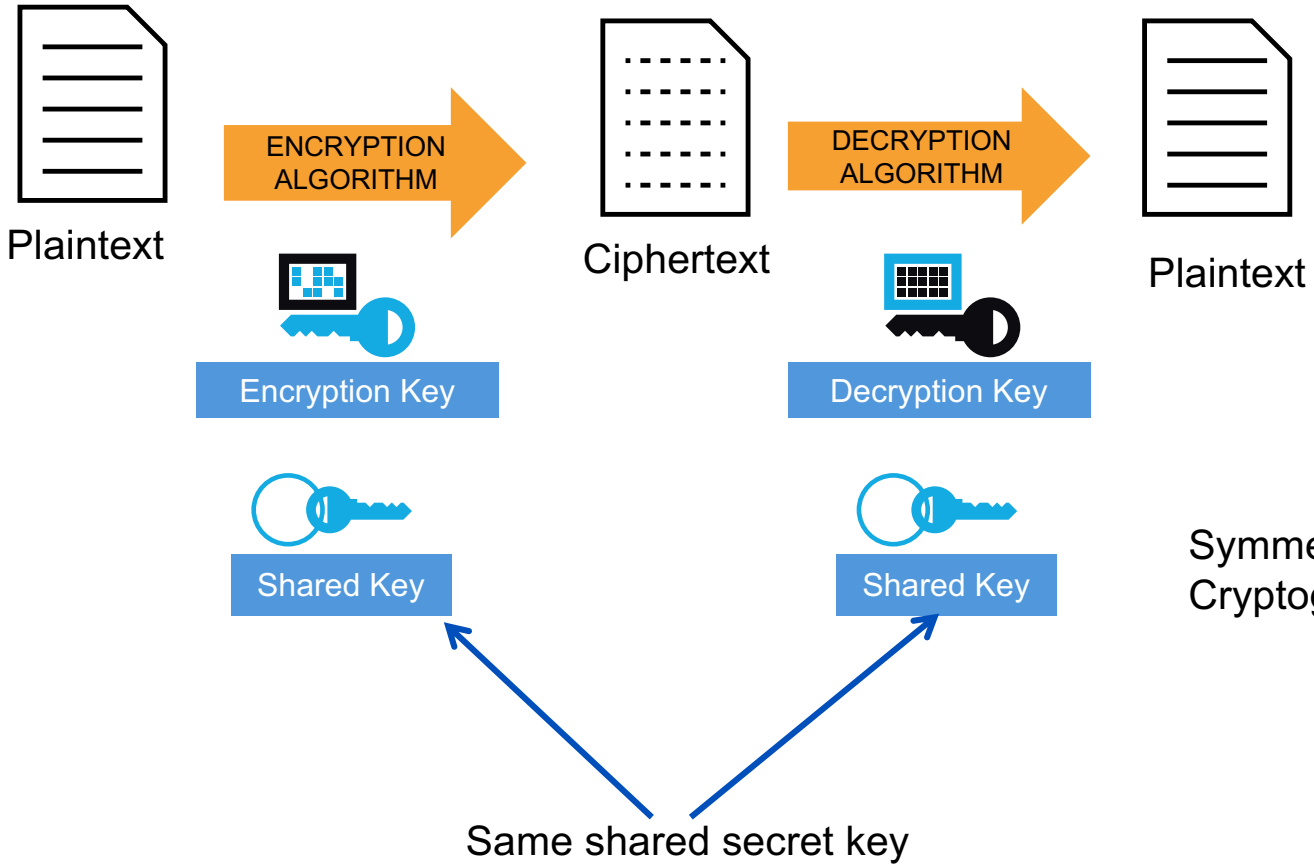
Encryption and Decryption



Symmetric Key Algorithm

- Uses a single key to both encrypt and decrypt information
- Also known as a secret-key algorithm
  - The key must be kept a “secret” to maintain security
  - This key is also known as a private key
- Follows the more traditional form of cryptography with key lengths ranging from 40 to 256 bits.
- Examples:
  - DES, 3DES, AES, RC4, RC6, Blowfish

Symmetric Encryption



Symmetric Key Cryptography

Symmetric Key Algorithm

Symmetric Algorithm	Key Size
DES	56-bit keys
Triple DES (3DES)	112-bit and 168-bit keys
AES	128, 192, and 256-bit keys
IDEA	128-bit keys
RC2	40 and 64-bit keys
RC4	1 to 256-bit keys
RC5	0 to 2040-bit keys
RC6	128, 192, and 256-bit keys
Blowfish	32 to 448-bit keys

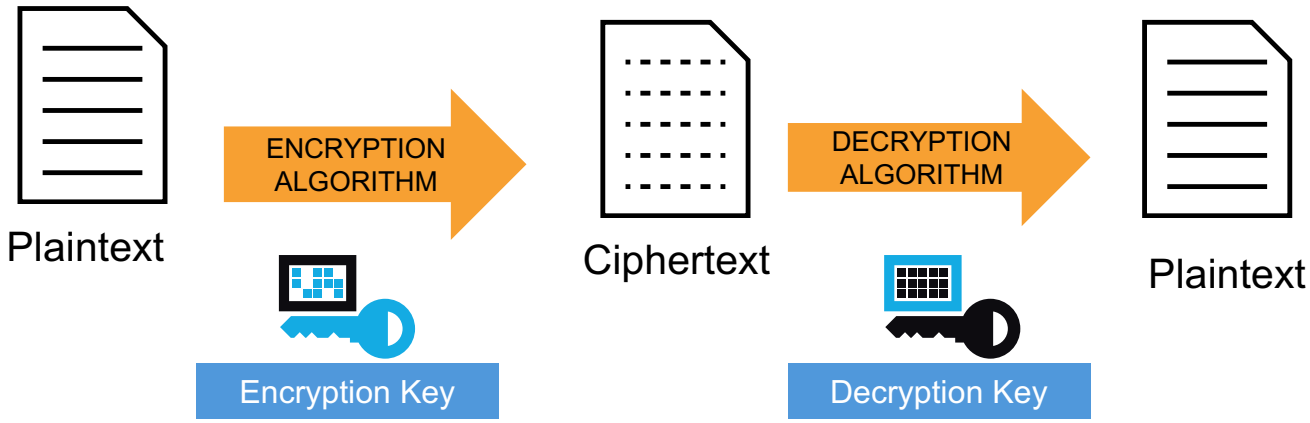
Note:
Longer keys are more difficult to crack, but more computationally expensive.

Data Encryption Standard (DES)

- Developed by IBM for the US government in 1973-1974, and approved in Nov 1976.
- Based on Horst Feistel's Lucifer cipher
- block cipher using shared key encryption, 56-bit key length
- Block size: 64 bits

DES: Illustration

64-bit blocks of input text



56-bit keys +
8 bits parity

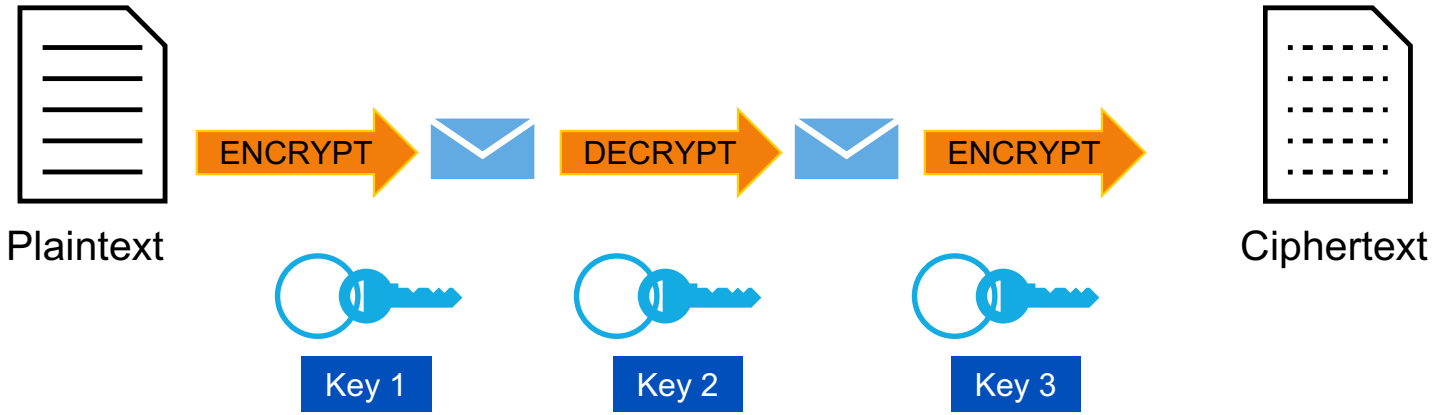
Triple DES

- 3DES (Triple DES) – a block cipher that applies DES three times to each data block
- Uses a key bundle comprising of three DES keys (K1, K2, K3), each with 56 bits excluding parity.
- DES encrypts with K1, decrypts with K2, then encrypts with K3

$$C_i = E_{K3}(D_{K2}(E_{K1}(P_i)))$$

- Disadvantage: very slow

3DES: Illustration



- Note:
  - If Key1 = Key2 = Key3, this is similar to DES
  - Usually, Key1 = Key3

Advanced Encryption Standard (AES)

- Published in November 2001
- Symmetric block cipher
- Has a fixed block size of 128 bits
- Has a key size of 128, 192, or 256 bits
- Based on Rijndael cipher which was developed by Joan Daemen and Vincent Rijmen
- Better suited for high-throughput, low latency environments

Rivest Cipher

RC Algorithm	Description
RC2	Variable key-sized cipher used as a drop in replacement for DES
RC4	Variable key sized stream cipher; Often used in file encryption and secure communications (SSL)
RC5	Variable block size and variable key length; uses 64-bit block size; Fast, replacement for DES
RC6	Block cipher based on RC5, meets AES requirement

- Chosen for speed and variable-key length capabilities
- Designed mostly by Ronald Rivest
- Each of the algorithms have different uses

Block Cipher

- Transforms a fixed-length block of plain text into a block of ciphertext
  - operate on a pre-determined block of bits (one byte, one word, 512 bytes, so forth), mixing key data in with the message data in a variety of different ways
- Common block ciphers:
  - DES and 3DES (in ECB and CBC mode)
  - Skipjack
  - Blowfish
  - RSA
  - AES
  - IDEA
  - SAFER

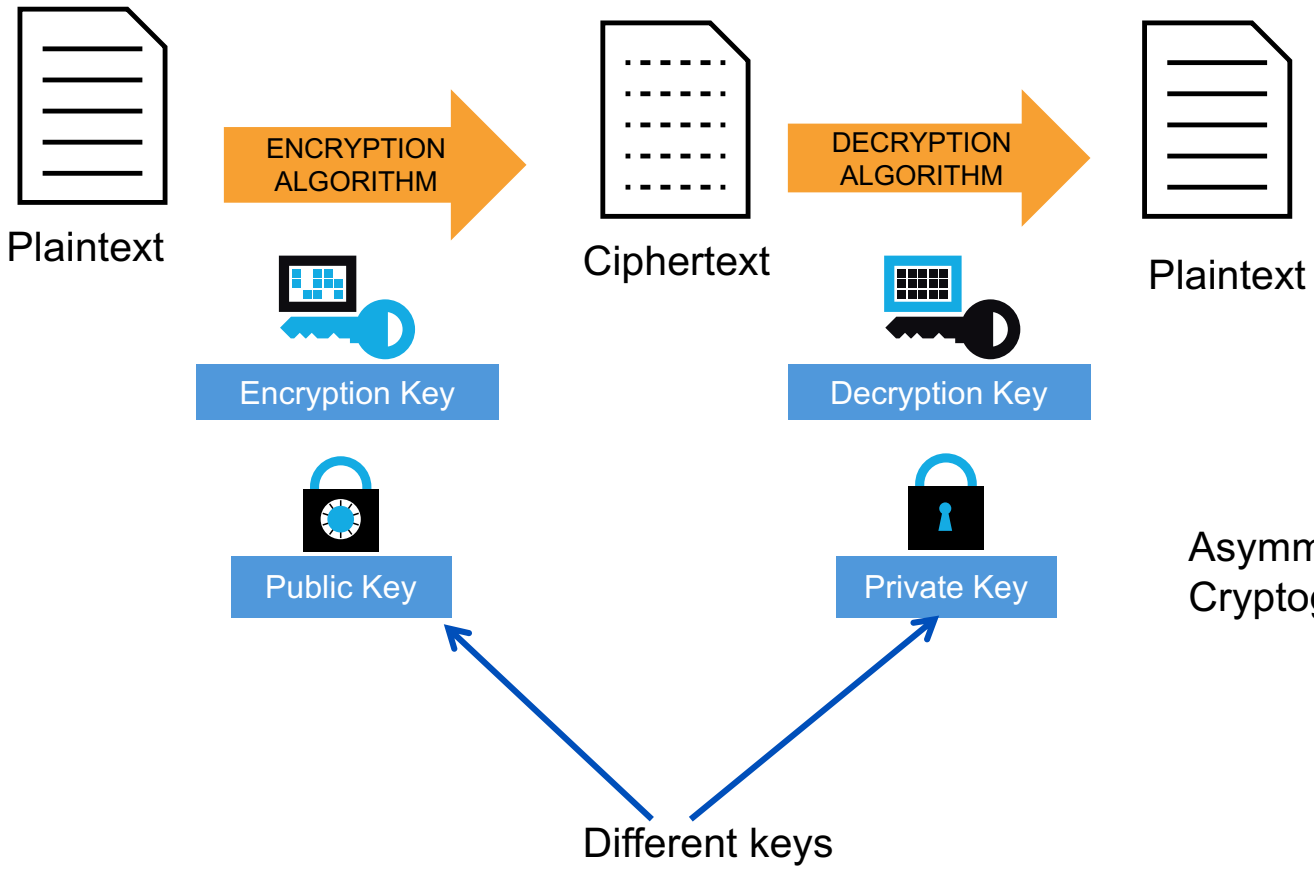
Stream Cipher

- Use smaller units of plaintext than what are used with block ciphers.
  - encrypts bits of the message at a time
  - typically bit-wise
  - They perform some operation (typically an exclusive OR) with one of these key bits and one of the message bits
  - They either have a very long key (that eventually repeats) or a reusable key that generates a repeatable but seemingly random string of bits.
- Common stream ciphers:
  - RC4
  - DES and 3DES (running OFB or CFB mode)
  - SEAL

Asymmetric Key Algorithm

- Also called public-key cryptography
  - Keep private key private
  - Anyone can see public key
- separate keys for encryption and decryption (public and private key pairs)
- Examples:
  - RSA, DSA, Diffie-Hellman, ElGamal, PKCS

Asymmetric Encryption



Asymmetric Key Cryptography

Asymmetric Key Algorithm

- RSA – the first and still most common implementation
- DSA – specified in NIST's Digital Signature Standard (DSS), provides digital signature capability for authentication of messages
- Diffie-Hellman – used for secret key exchange only, and not for authentication or digital signature
- ElGamal – similar to Diffie-Hellman and used for key exchange
- PKCS – set of interoperable standards and guidelines

Symmetric vs. Asymmetric Key

Symmetric	Asymmetric
generally fast Same key for both encryption and decryption	Can be 1000 times slower Uses two different keys (public and private) Decryption key cannot be calculated from the encryption key Key lengths: 512 to 4096 bits Used in low-volume

Hash Functions

- produces a condensed representation of a message
- takes an input message of arbitrary length and outputs fixed-length code
  - The fixed-length output is called the hash or message digest
- A form of signature that uniquely represents the data
- Uses:
  - Verifying file integrity
  - Digitally signing documents
  - Hashing passwords

Hash Functions

- Message Digest (MD) Algorithm
  - Outputs a 128-bit fingerprint of an arbitrary-length input
  - MD4 is obsolete, MD5 is widely-used

- Secure Hash Algorithm (SHA)
  - SHA-1 produces a 160-bit message digest similar to MD5
  - Widely-used on security applications (TLS, SSL, PGP, SSH, S/MIME, IPsec)
  - SHA-256, SHA-384, SHA-512 can produce hash values that are 256, 384, and 512-bits respectively

Digital Signature

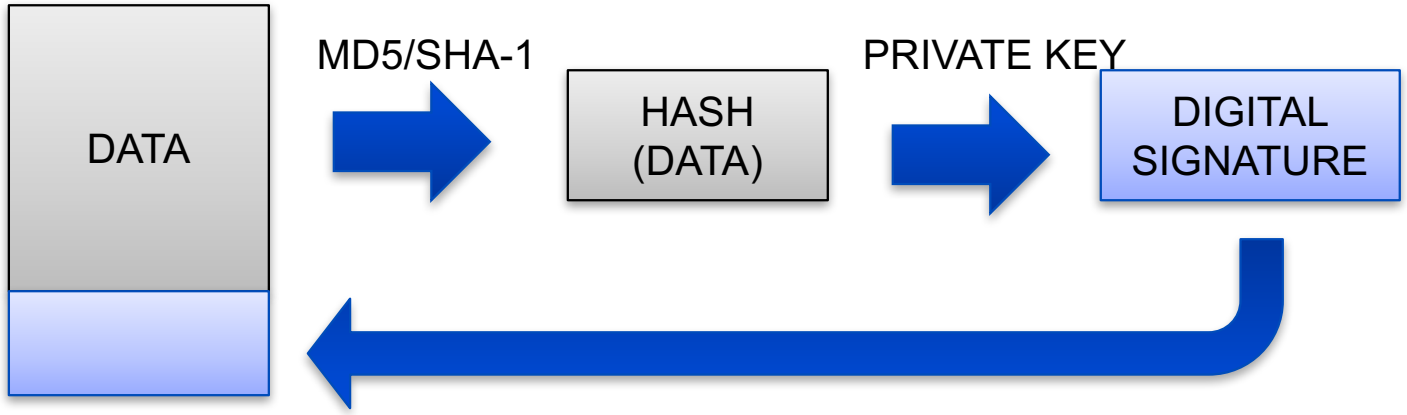
- A digital signature is a message appended to a packet
- The sender encrypts message with own private key instead of encrypting with intended receiver's public key
- The receiver of the packet uses the sender's public key to verify the signature.
- Used to prove the identity of the sender and the integrity of the packet

Digital Signature

- a message appended to a packet
- used to prove the identity of the sender and the integrity of the packet
- how it works:
  - sender signs the message with own private key
  - receiver uses the sender's public key to verify the signature

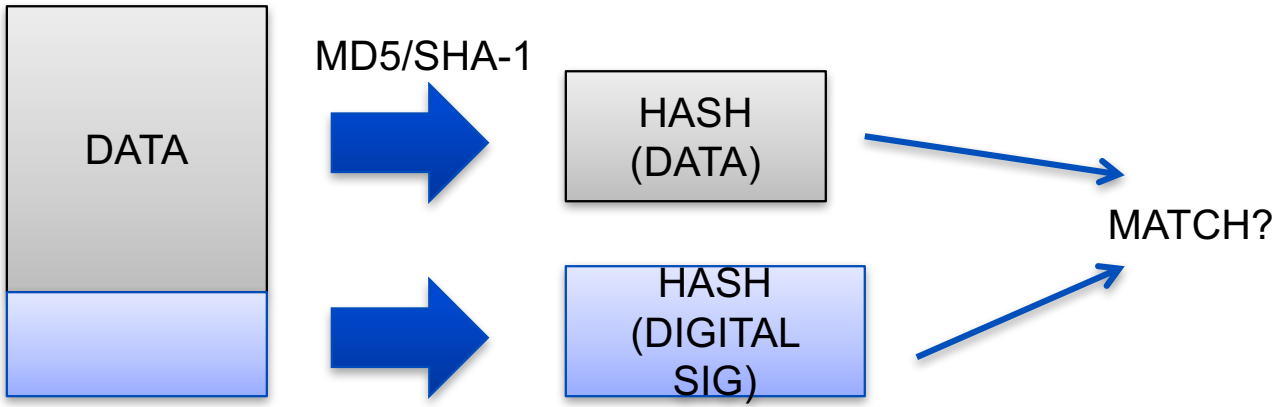
Digital Signature Process

- Hash the data using one of the supported hashing algorithms (MD5, SHA-1, SHA-256)
- Encrypt the hashed data using the sender's private key
- Append the signature (and a copy of the sender's public key) to the end of the data that was signed)



Signature Verification Process

- Hash the original data using the same hashing algorithm
- Decrypt the digital signature using the sender's public key. All digital signatures contain a copy of the signer's public key
- Compare the results of the hashing and the decryption. If the values match then the signature is verified. If the values do not match, then the data or signature was probably modified.



Message Authentication Code

- Provides integrity and authenticity
- How it works:
  - In the sender side, the message is passed through a MAC algorithm to get a MAC (or Tag)
  - In the receiver side, the message is passed through the same algorithm
  - The output is compared with the received tag and should match
- Uses the same secret key
- Can also use hash function to generate the MAC, called Hash-based Message Authentication Code (HMAC)


Questions

APNIC 44

Public Key Infrastructure

TAICHUNG, TAIWAN

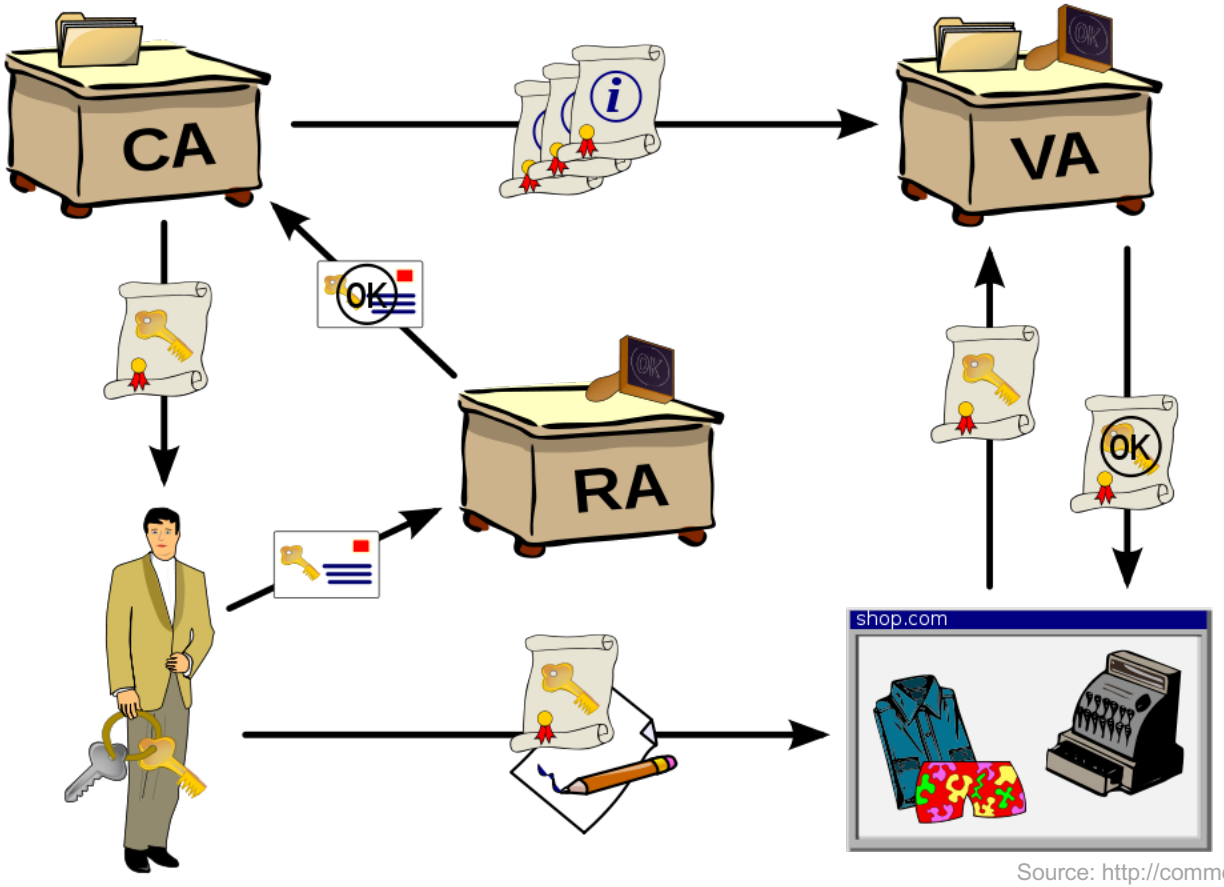
7–14 September 2017

#apnic44

Public Key Infrastructure

- Framework that builds the network of trust
- Combines public key cryptography, digital signatures
- Ensures confidentiality, integrity, authentication, nonrepudiation, and access control
- Protects applications that require high level of security

Public Key Infrastructure



Source: http://commons.wikimedia.org

Functions of a PKI

- Registration
- Initialization
- Certification
- Key pair recovery
- Key generation
- Key update
- Cross-certification
- Revocation

Components of a PKI

- Certificate authority
  - The trusted third party
  - Trusted by both the owner of the certificate and the party relying upon the certificate.
- Validation authority
- Registration authority
  - For big CAs, a separate RA might be necessary to take some work off the CA
  - Identity verification and registration of the entity applying for a certificate
- Central directory

Certificate Authority

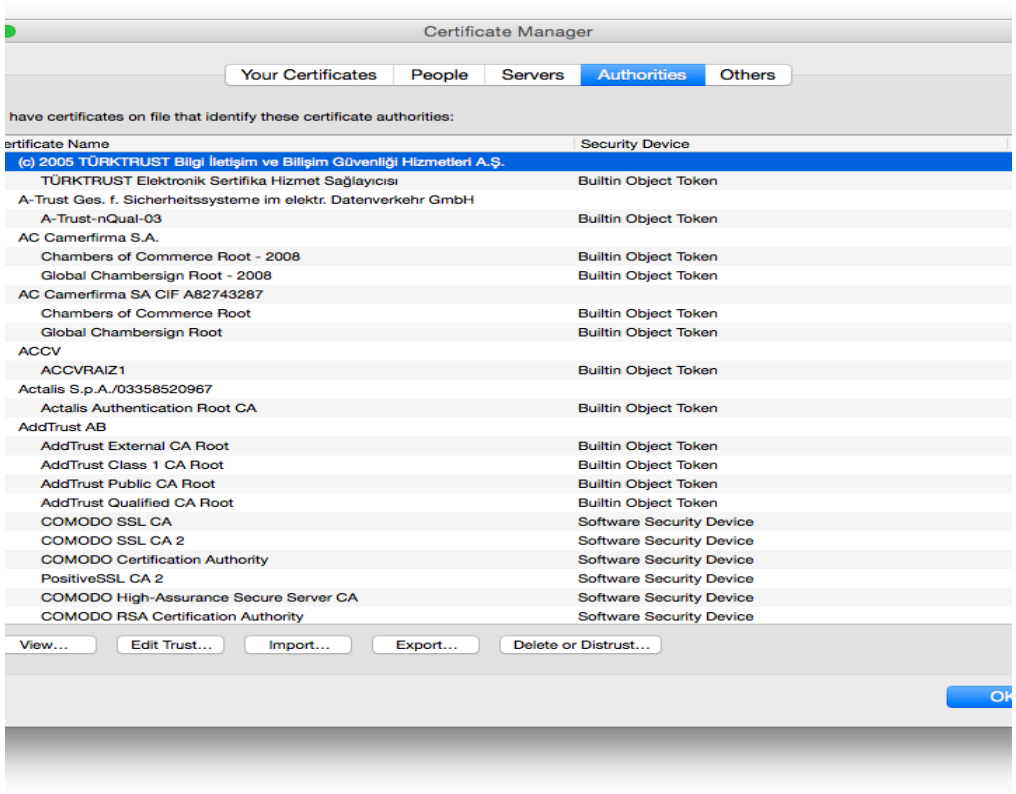
- Issuer and signer of the certificate
- Trusted (Third) Party
  - Based on trust model
  - Who to trust?
- Types:
  - Enterprise CA
  - Individual CA (PGP)
  - Global CA (such as VeriSign)
- Functions:
  - Enrolls and Validates Subscribers
  - Issues and Manages Certificates
  - Manages Revocation and Renewal of Certificates
  - Establishes Policies & Procedures

Who decides a CA can be trusted?

- Your browser, operating system, and mobile devices
- CA membership programs
  - If CA meets criteria, the CA issues certificates that are then trusted by these devices
- Browsers trust the Root Certificate and keeps in a database of approved CAs called the root store

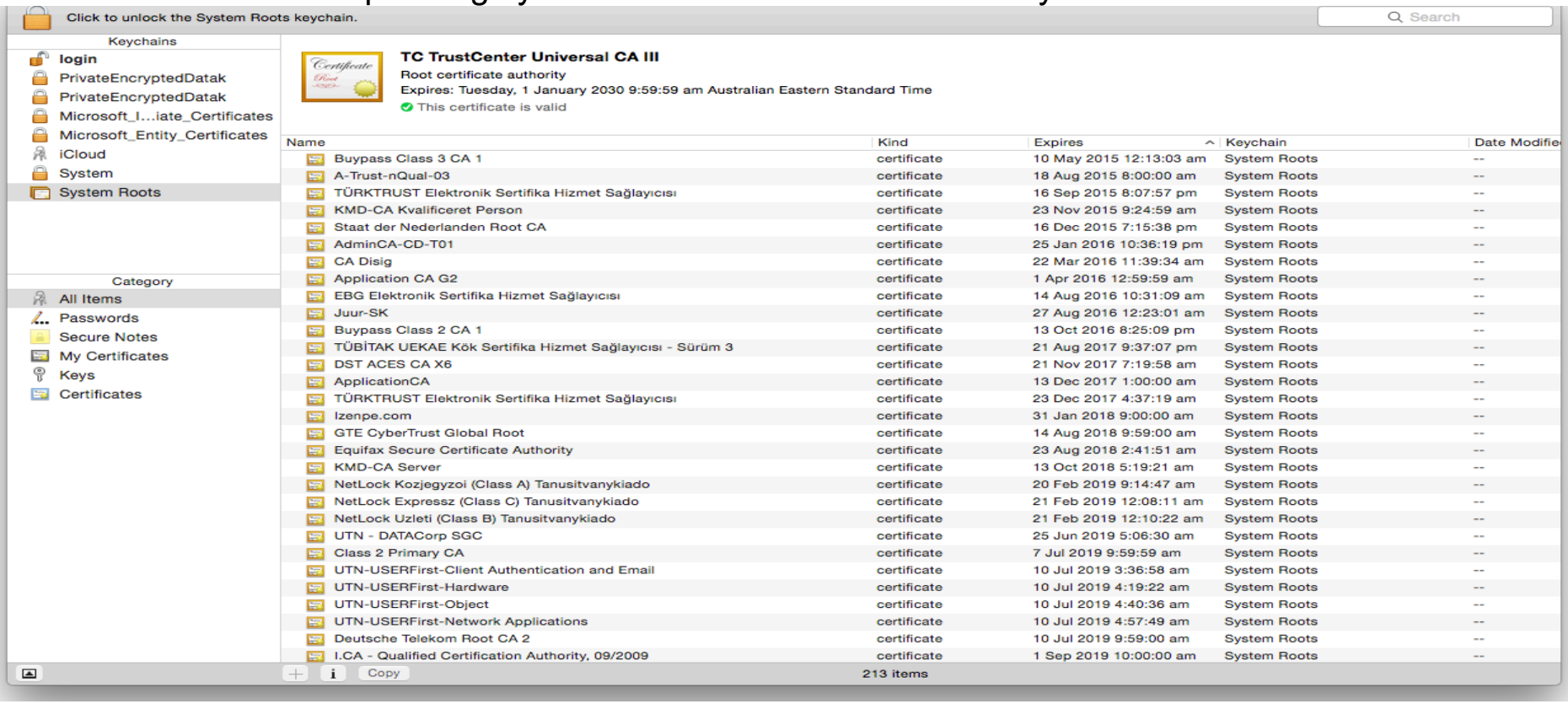
Root CAs

- Your browser trusts 500+ root CAs by default
- In Firefox:
  - Preferences > Advanced > Certificates > View Certificates > Authorities



Root CAs

Your operating system trusts a few hundred CAs by default



Certificates

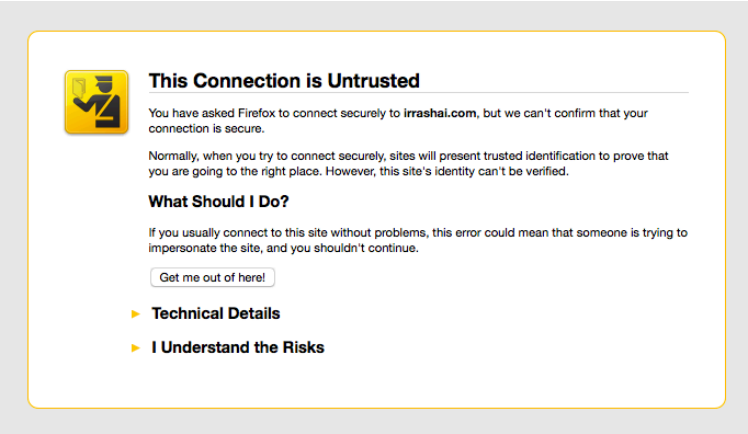
- Public key certificates bind public key values to subjects
- A trusted certificate authority (CA) verifies the subject's identity and digitally signs each certificate
  - Validation
- Has a limited valid lifetime
- Can be used using untrusted communications and can be cached in unsecured storage
  - Because client can independently check the certificate's signature
- Certificate is NOT equal to signature
  - It is implemented using signature
- Certificates are static
  - If there are changes, it has to be re-issued

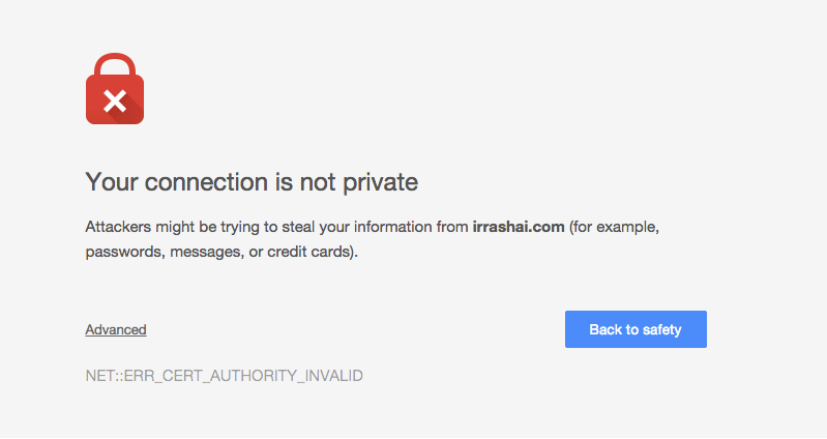
Digital Certificate

- Digital certificate – basic element of PKI; secure credential that identifies the owner
- Also called public key certificate
- Certificate examples:
  - X509 (standard)
  - PGP (Pretty Good Privacy)

How to Verify Certificate

- Most browsers give warning
  - “This connection is untrusted”
- Up to you if you want to proceed
  - Add exception





Public CA – Let’s Encrypt

- A community-driven certificate authority
- “Let’s Encrypt” aims to automate the process of obtaining and installing, and maintaining a secure website.
- It is also provided for free
- Written on Boulder

https://letsencrypt.org/

Every certificate contains…

- Body of the certificate
  - Version number, serial number, names of the issuer and subject
  - Public key associated with the subject
  - Expiration date (not before, not after)
  - Extensions for additional tributes
- Signature algorithm
  - Used by the CA to sign the certificate
- Signature
  - Created by applying the certificate body as input to a one-way hash function. The output value is encrypted with the CA's private key to form the signature value

Digital Certificate

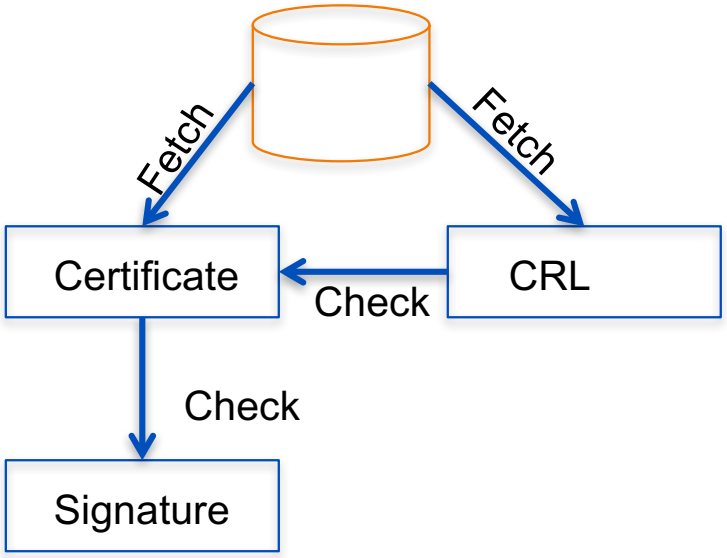
- To obtain a digital certificate, Alice must:
  - Make a certificate signing request to the CA
  - Alice sends to CA:
    - Her identifier IdA
    - Her public key KA_PUB
    - Additional information
- CA returns Alice's digital certificate, cryptographically binding her identity to public key:
  - CertA = {IDA, KA_PUB, info, SigCA(IDA,KA_PUB,info)}

X.509

- An ITU-T standard for a public key infrastructure (PKI) and Privilege Management Infrastructure (PMI)
- Assumes a strict hierarchical system of Certificate Authorities (CAs)
- RFC 1422 – basis of X.509-based PKI
- Current version X.509v3 provides a common baseline for the Internet
- Structure of a Certificate, certificate revocation (CRLs)

X.509 Certificate Usage

- Fetch certificate
- Fetch certificate revocation list (CRL)
- Check the certificate against the CRL
- Check signature using the certificate



Certificate Revocation Lists

- CA periodically publishes a data structure called a certificate revocation list (CRL).
- Described in X.509 standard.
- Each revoked certificate is identified in a CRL by its serial number.
- CRL might be distributed by posting at known Web URL or from CA's own X.500 directory entry.


Questions

APNIC 44

Pretty Good Privacy

TAICHUNG, TAIWAN

7–14 September 2017

#apnic44

Pretty Good Privacy

- Created by Phil Zimmerman in 1991 originally using symmetric encryption
- PGP 3 allowed for asymmetric encryption
- Zimmerman's team and Viacrypt (who'd licensed RSA from RSADSI) merged to form PGP Inc in 1996
- OpenPGP as a standard proposed to IETF in 1997 to avoid patent issues.
- PGP Inc now owned by Symantec
- GPG is the Free Software Foundation's implementation of the OpenPGP standard

Signing and Encrypting

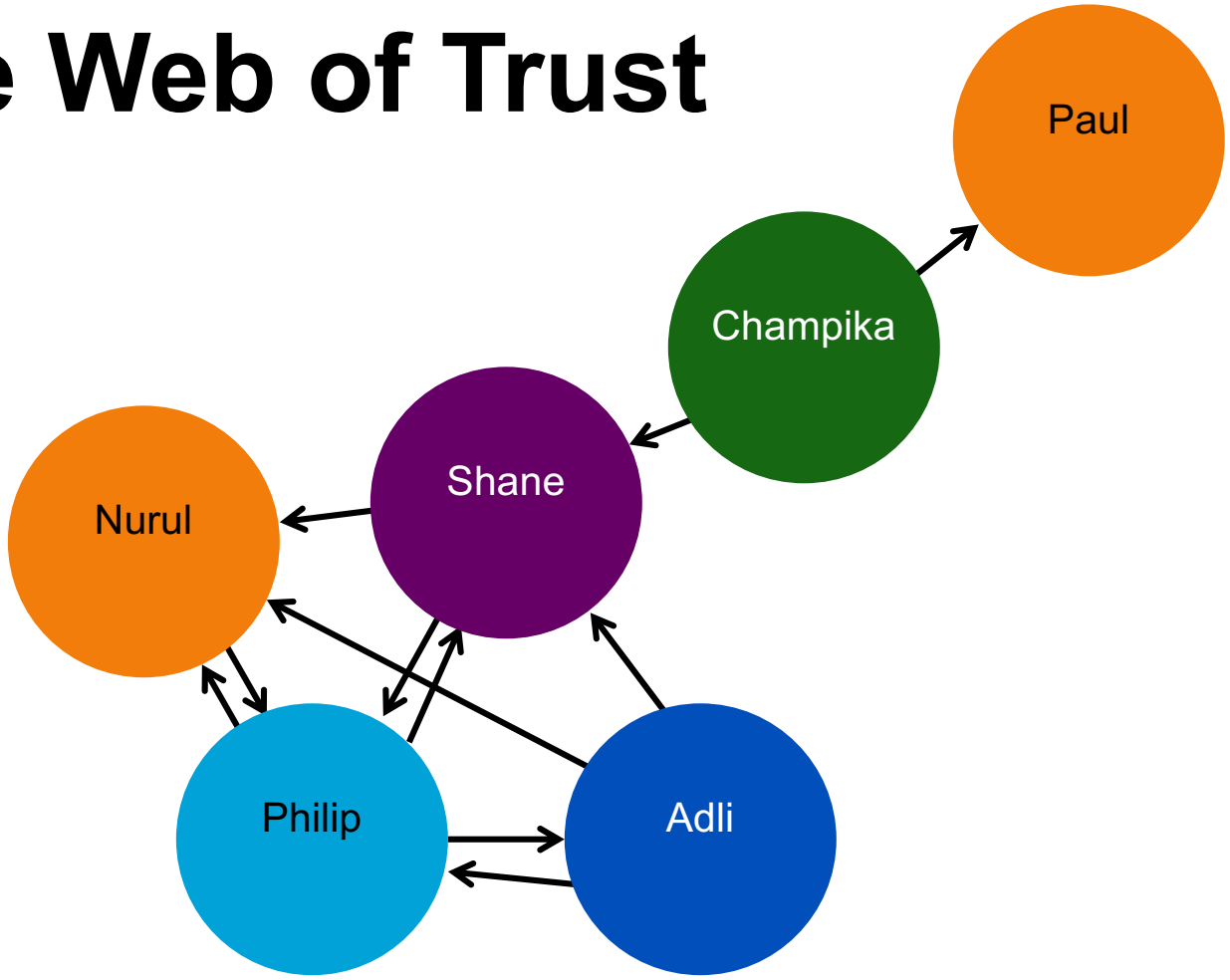
- Data is encrypted with a public key to be decrypted with the corresponding private key.
- Data can be signed with the private key to be verified by anyone who has the corresponding public key.
- Since public keys are data, they can be signed too.
- Hash functions that generate fixed length fingerprints of any input data can be used to identify keys that would otherwise be over 1024 bits long

Trust

- Centralized / hierarchal trust – where certain globally trusted bodies sign keys for every one else.
- Decentralized webs of trust – where you pick who you trust yourself, and decide if you trust who those people trust in turn.

Which works better for what reasons?

Sample Web of Trust



Installing GnuPG Software

- GNU Privacy Guard
- Core software either commercial from pgp or opensource from gnupg
  - GPG4Win for windows
  - GPGTools for Mac OS X
  - Your package manager for Linux/UNIX
  - Source code from https://www.gnupg.org/
- Written by Werner Koch in 1997
  - After attending a talk by Stallman

Installing GnuPG Software

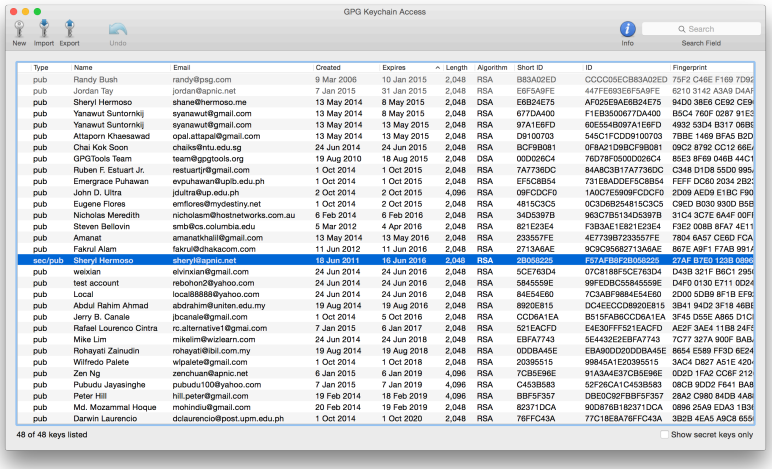
- Core software either commercial from pgp or opensource from gnupg.
- https://www.gpg4win.org/ for windows
- https://www.gpgtools.org/ for OS X
- Your package manager for Linux/UNIX
- Source code from https://www.gnupg.org/

Key management: generation

- Using graphical tools based on what you installed above:
  - GPG Keychain Access for OS X
  - Kleopatra or GPA for windows
- Using the command line:
  - gpg --gen-key
- Generate a key – use your email address. The comment field can be left blank.

Key Management

- Using graphical tools, you may see your own keypair, and all imported public keys from contacts
  - Keychain Access (Mac OS X)
  - Kleopatra (Windows)



Type	Name	Email	Created	Expires	Length	Algorithm	Short ID	ID	Fingerprint
pub	Randy Bush	randy@psg.com	9 Mar 2006	10 Jan 2015	2,048	RSA	B83A02ED	CCCC05ECB83A02ED	75F2 C46E F169 7D92
pub	Jordan Tay	jordan@apnic.net	7 Jan 2015	31 Jan 2015	2,048	RSA	E6F5A9FE	447FE693E6F5A9FE	6210 3142 A3A9 D4AF
pub	Sheryl Hermoso	shane@hermoso.me	13 May 2014	8 May 2015	2,048	DSA	E6B24E75	AF025E9AE6B24E75	94D0 38E6 CE92 CE9[illegible]
pub	Yanawut Suntornkij	syanawut@gmail.com	13 May 2014	8 May 2015	2,048	RSA	677DA400	F1EB350067 7DA400	B5C4 760F 0287 91E[illegible]
pub	Yanawut Suntornkij	syanawut@gmail.com	13 May 2014	13 May 2015	2,048	RSA	97A1E6FD	60E55480 97A1E6FD	4932 53D4 B317 06B[illegible]
pub	Attaporn Khaesawad	opal.attapal@gmail.com	13 May 2014	13 May 2015	2,048	RSA	D9100703	545C1FCDD9100703	7BBE 1469 BFA5 B2D[illegible]
pub	Chai Kok Soon	chaiks@ntu.edu.sg	24 Jun 2014	24 Jun 2015	2,048	RSA	BCF9B081	0F8A21D9BCF9B081	09C2 8792 CC12 66E[illegible]
pub	GPGTools Team	team@gpgtools.org	19 Aug 2010	18 Aug 2015	2,048	DSA	00D026C4	76D78F0500D026C4	85E3 8F69 046B 44C1
pub	Ruben F. Estuart Jr.	restuartjr@gmail.com	1 Oct 2014	1 Oct 2015	2,048	RSA	7A7736DC	84A8C3B17A7736DC	C348 D1D8 55D0 995[illegible]
pub	Emergrace Puhawan	expuhawan@upib.edu.ph	1 Oct 2014	1 Oct 2015	2,048	RSA	EF5C8B54	731E8ADDEF5C8B54	FEFF DC60 2034 2B2[illegible]
pub	John D. Ultra	jdultra@up.edu.ph	2 Oct 2014	2 Oct 2015	4,096	RSA	09FCDCF0	1A0C7E5909FCDCF0	2D09 AED9 E1BC F90[illegible]
pub	Eugene Flores	emflores@mydestiny.net	1 Oct 2014	2 Oct 2015	2,048	RSA	4815C3C5	0C3D6B254815C3C5	C9ED B030 930D B5E[illegible]
pub	Nicholas Meredith	nicholasm@hostnetworks.com.au	6 Feb 2014	6 Feb 2016	2,048	RSA	34D5397B	963C7B5134D5397B	31C4 3C7E 6A4F 00F[illegible]
pub	Steven Bellovin	smb@cs.columbia.edu	5 Mar 2012	4 Apr 2016	2,048	RSA	821E23E4	F3B3AE1E821E23E4	F3E2 00B8 8FA7 4E1[illegible]
pub	Amanat	amanatkhalil@gmail.com	13 May 2014	13 May 2016	2,048	RSA	233557FE	4E7739B7233557FE	7804 8A57 CE6D FCA[illegible]
pub	Fakrul Alam	fakrul@dhakacom.com	11 Jun 2012	11 Jun 2016	2,048	RSA	2713A6AE	9C9C95682713A6AE	867E A9F1 F7AB 991[illegible]
sec/pub	Sheryl Hermoso	sheryl@apnic.net	16 Jun 2011	16 Jun 2016	2,048	RSA	2B058225	F57AFB8F2B058225	27AF B7E0 123B 0896
pub	weixian	elvinxian@gmail.com	24 Jun 2014	24 Jun 2016	2,048	RSA	5CE763D4	07C8188F5CE763D4	D43B 321F B6C1 295[illegible]
pub	test account	rebohon2@yahoo.com	24 Jun 2014	24 Jun 2016	2,048	RSA	5845559E	99FEDBC55845559E	D4F0 0130 E711 0D2[illegible]
pub	Local	local88888@yahoo.com	24 Jun 2014	24 Jun 2016	2,048	RSA	84E54E60	7C3ABF9884E54E60	2D00 5DB9 8F1B EF9[illegible]
pub	Abdul Rahim Ahmad	abdrahim@uniten.edu.my	19 Aug 2014	19 Aug 2016	2,048	RSA	8920E815	DC4EECCDB920E815	3B41 94D2 3F18 46BE
pub	Jerry B. Canale	jbcanale@gmail.com	1 Oct 2014	5 Oct 2016	2,048	RSA	CCD6A1EA	B515FAB6CCD6A1EA	3F45 D55E A865 D1C[illegible]
pub	Rafael Lourenco Cintra	rc.alternative1@gmai.com	7 Jan 2015	6 Jan 2017	2,048	RSA	521EACFD	E4E30FFF521EACFD	AE2F 3AE4 11B8 24F[illegible]
pub	Mike Lim	mikelim@wizlearn.com	24 Jun 2014	24 Jun 2018	2,048	RSA	EBFA7743	5E4432E2EBFA7743	7C77 327A 900F 8AB[illegible]
pub	Rohayati Zainudin	rohayati@ibil.com.my	19 Aug 2014	19 Aug 2018	2,048	RSA	0DDBA45E	EBA90D0D0DBA45E	8654 E589 FF3D BE24
pub	Wilfredo Palete	wlpalete@gmail.com	1 Oct 2014	1 Oct 2018	2,048	RSA	20395515	99845A1E20395515	3AC4 D827 A51E 420[illegible]
pub	Zen Ng	zenchuan@apnic.net	6 Jan 2015	6 Jan 2019	4,096	RSA	7CB5E96E	91A3A4E37CB5E96E	0D2D 1FA2 CC6F 212[illegible]
pub	Pubudu Jayasinghe	pubudu100@yahoo.com	7 Jan 2015	7 Jan 2019	4,096	RSA	C453B583	52F26CA1C453B583	08CB 9DD2 F641 BA8[illegible]
pub	Peter Hill	hill.peter@gmail.com	19 Feb 2014	18 Feb 2019	4,096	RSA	BBF5F357	DBE0C92FBBF5F357	28A2 C980 84DB 4A8[illegible]
pub	Md. Mozammal Hoque	mohindiu@gmail.com	20 Feb 2014	19 Feb 2019	2,048	RSA	82371DCA	90D876B182371DCA	0896 25A9 EDA3 1B3[illegible]
pub	Darwin Laurencio	dclaurencio@post.upm.edu.ph	1 Oct 2014	1 Oct 2020	2,048	RSA	76FFC43A	77C16E8A76FFC43A	3B2B 4EA5 A9C8 655[illegible]

Key management: distribution

- On printed media: published book or business cards:
- Digitally in email or using sneaker-net
- Online using the openpgp key servers.
- Still does not tell you if you trust the key.

Key management: rollover

- Expiry dates ensure that if your private key is compromised they can only be used till they expire.
- Can be changed after creating the key.
- Before expiry, you need to create a new key, sign it with the old one, send the signed new one to everyone in your web of trust asking them to sign your new key.
- Many people create keys that don’t expire. Think about the security implications of that.

Key management: revocation

- Used to mark a key as invalid before its expiry date.
- Always generate a revocation certificate as soon as you create your key.
- Do not keep your revocation certificate with your private key.
  - gpg --gen-revoke IDENTITY

Key management: partying

- Key signing parties are ways to build webs of trust.
- Each participant carries identification, as well as a copy of their key fingerprint. (maybe some $ as well ☺)
- Each participant decides if they’re going to sign another key based on their personal policy.
- Keys are easiest kept in a keyring on an openpgp keyserver in the aftermath of the party.

Interesting gpg commands

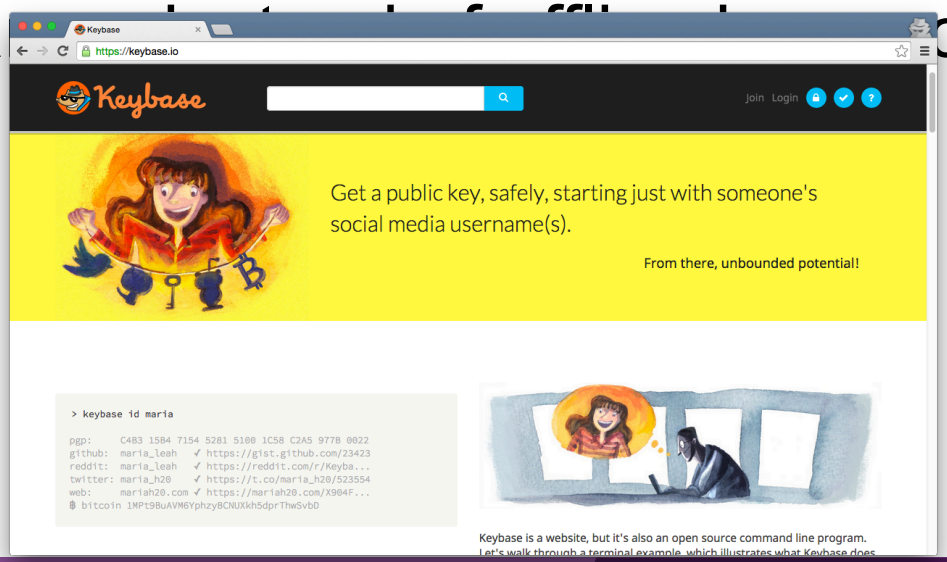
- Get help for gpg options
  - gpg --help AND man gpg
- Print the fingerprint of a particular key
  - gpg --fingerprint IDENTITY
- IDENTITY = email or PGP key ID
- Export a public key to an ASCII armored file.
  - gpg -a --output my-public-key.asc --export IDENTIY

Interesting gpg commands

- Import a key from a file into your keyring
  - gpg --import public.asc
- Import a key from a keyserver
  - gpg --recv-keys --keyserver hkp://keys.gnupg.net
- Send your key to a keyserver
  - gpg --send-keys --keyserver hkp://keys.gnupg.net
- Sign a key
  - gpg --sign-key IDENTITY

Keybase

- Uses PGP to encrypt and verify files, usernames, accounts, etc..
- Goal is to let any security software be powered by (social media) usern changes



APNIC 44

Thank you

TAICHUNG, TAIWAN

7–14 September 2017

#apnic44