[I05] solutions

[I05_t01] using cryptography

wim mees

introduction

information security triad

what are we protecting?

- **C**onfidentiality
- **I**ntegrity
- **A**vailability

in practice we often add:

- *authenticity:* the person who claims to be ‘someone’ is really that ‘someone’

basic principle

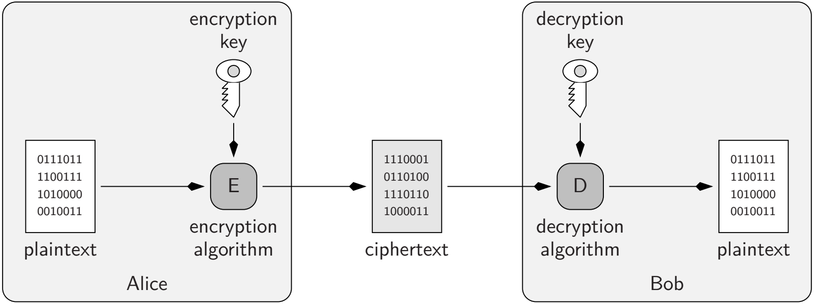


Figure 1: basic principle of cryptography

cryptography basics

types of cryptography

symmetric cryptography

a.k.a.

- *"secret key cryptography"*
- *"conventional cryptography"*

asymmetric cryptography

a.k.a.

- *"public key cryptography"*

symmetric cryptography

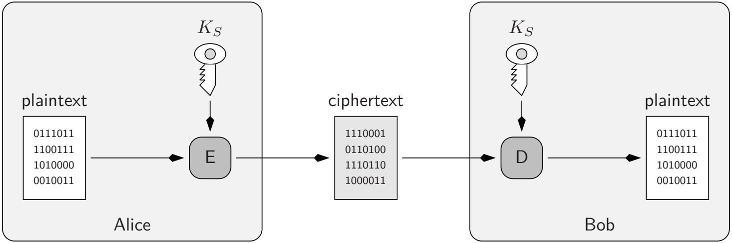


Figure 2: symmetric cryptography

- shared secret
- problem: key distribution

asymmetric cryptography

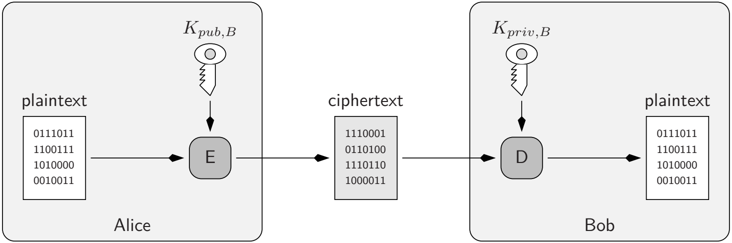


Figure 3: asymmetric cryptography for confidentiality

- each user generates a key-pair:
  - keeps one key secret, the *"private key"*
  - shares other key with the world, the *"public key"*
- wat is encrypted with one key from a key-pair, can only be decrypted with other key from same pair
- problem: message injection with spoofed sender

asymmetric cryptography

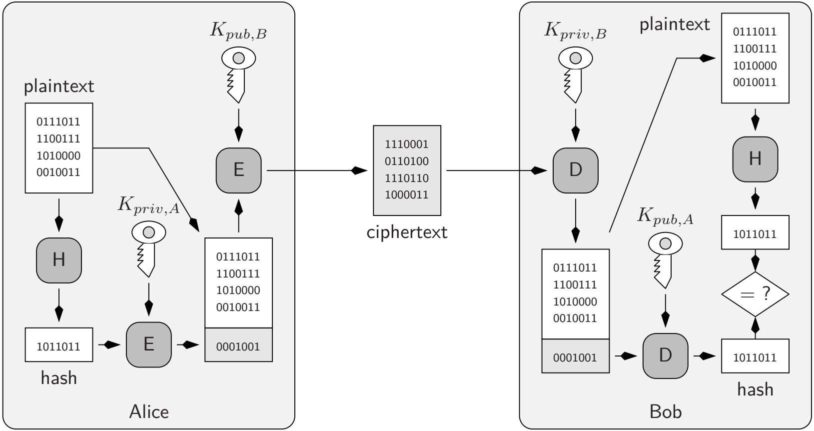


Figure 4: asymmetric cryptography with additional message signing

asymmetric cryptography

remaining threat

attacker

- implements *man-in-the-middle* attack between A and B
- creates fake key-pair for A and B
- replaces real public key by fake copy
  when A (or B) sends public key to B (or A)

conclusion

- still same problem of exchanging (in this case *public*) keys
- problem of *authenticity*...
  therefore the solution is ...?
  signing by a *trusted third party* !

public key infrastructure

centralized approach using a Certificate Authority (CA)

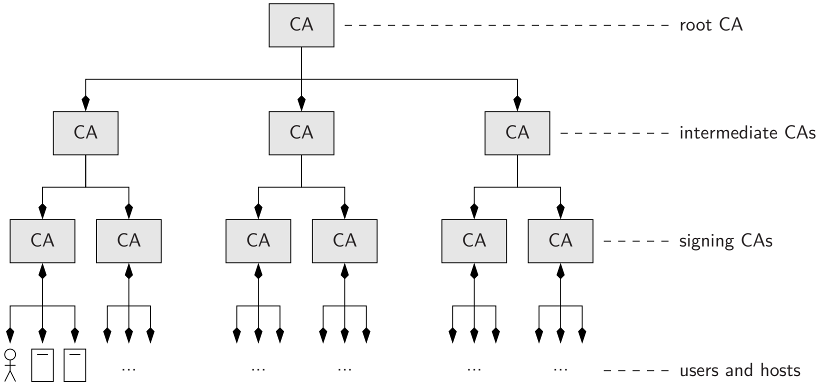


Figure 5: CA hierarchy

public key infrastructure

decentralized approach

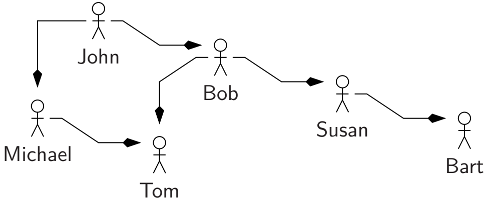


Figure 6: web of trust

network encryption

where to implement encryption

where is encryption implemented?

- *bump-in-the-stack*
- *bump-in-the-wire*

data link layer encryption

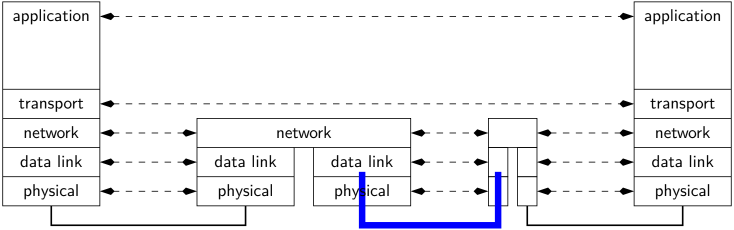


Figure 7: data link layer encryption

data link layer encryption

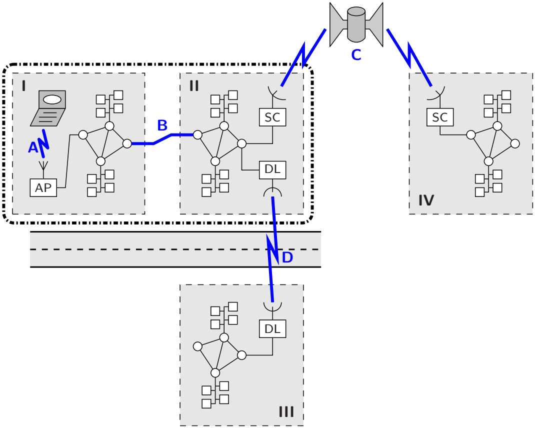


Figure 8: data link layer encryption use cases

data link layer encryption

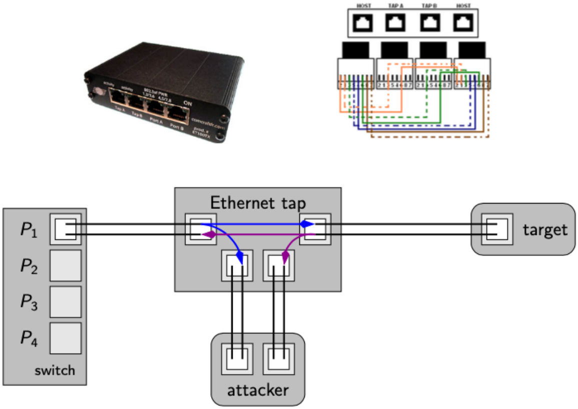
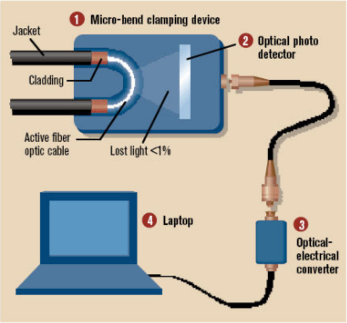
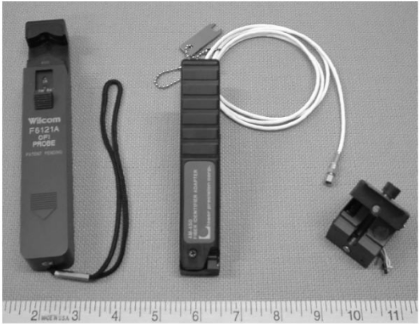


Figure 9: Ethernet tap

data link layer encryption



concept



actual tap hardware

Figure 10: fibre tap

data link layer encryption

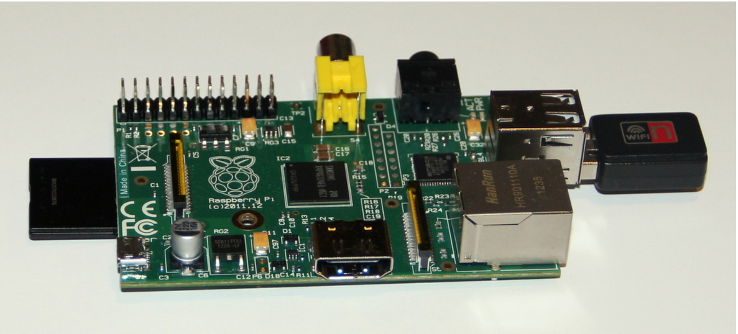


Figure 11: DIY: Raspberry Pi + WiFi dongle ≈ 30$

data link layer encryption

Figure 12: or buy a WiFi Pineapple Mark V: 100$

data link layer encryption

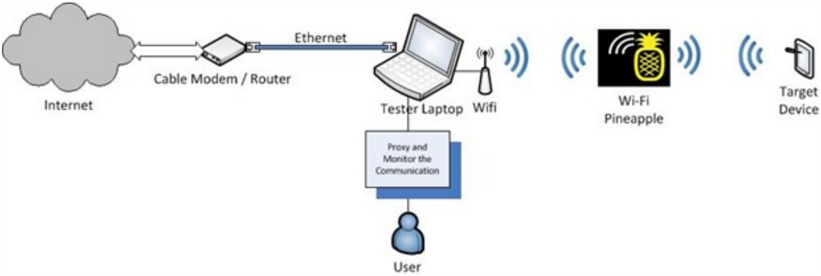


Figure 13: Pineapple Mark V

data link layer encryption

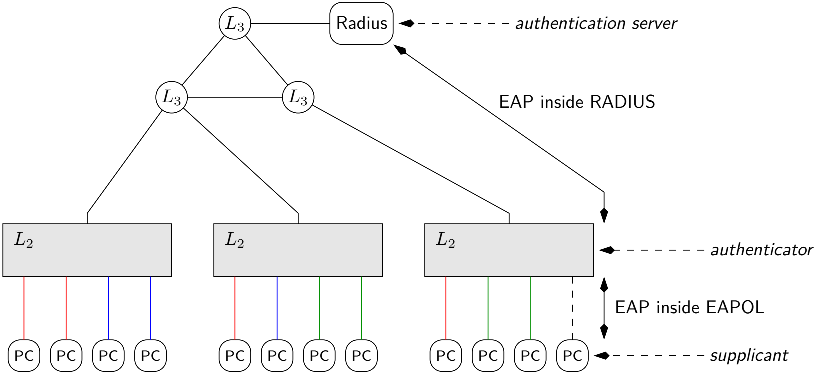


Figure 14: Network Access Control (NAC) 802.1x

data link layer encryption

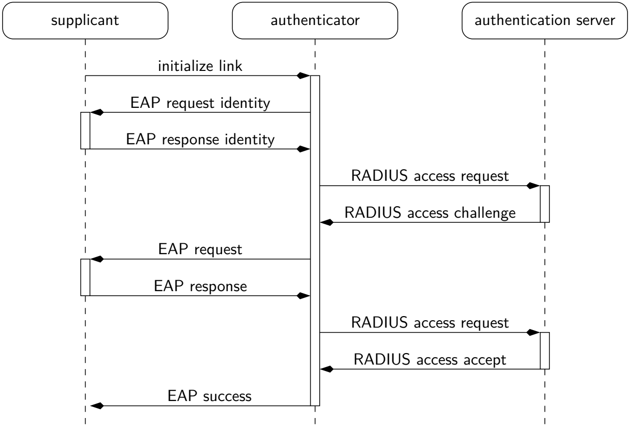


Figure 15: Network Access Control (NAC) 802.1x

data link layer encryption

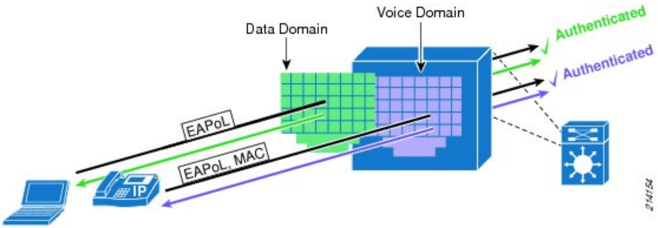


Figure 16: Cisco's *"Multi-Domain Authentication"* (MDA)

data link layer encryption

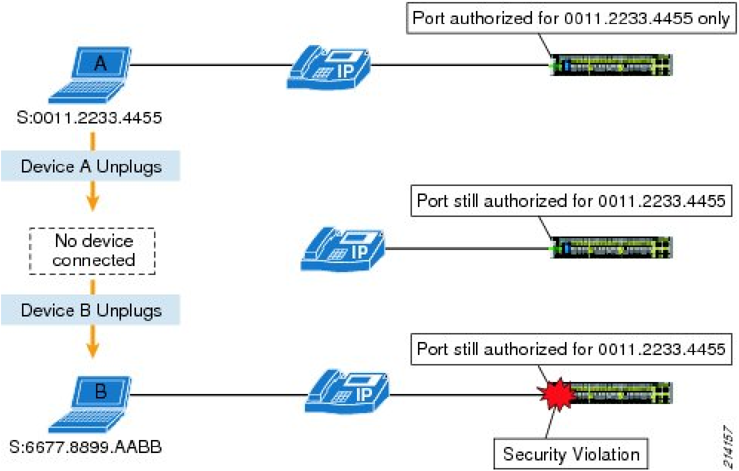


Figure 17: problem: device behind phone disconnects

data link layer encryption

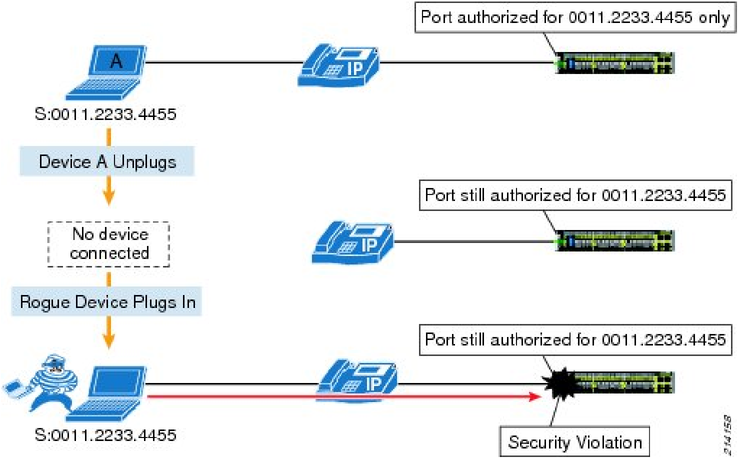


Figure 18: problem: device behind phone disconnects & MAC spoofing

data link layer encryption

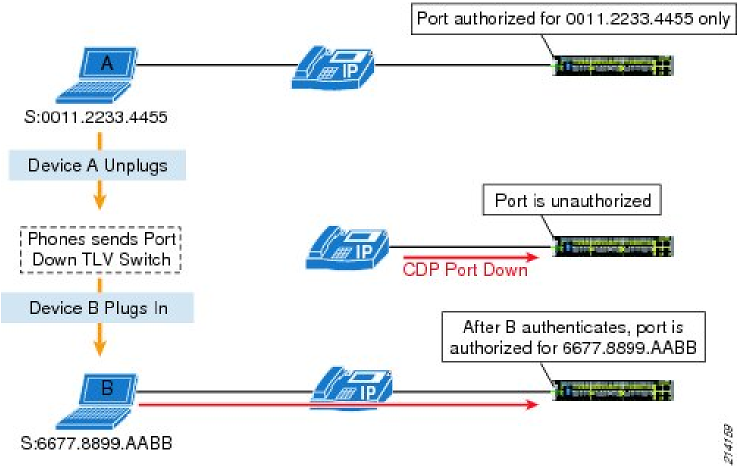


Figure 19: solution when available (otherwise . . . timers . . .)

network layer encryption

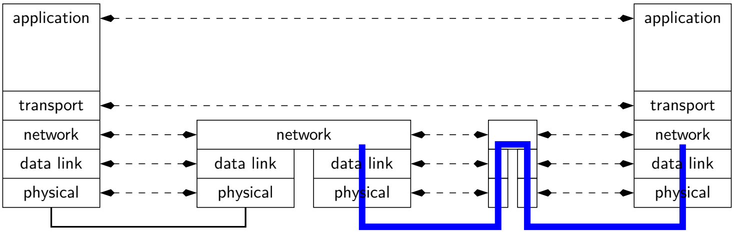


Figure 20: network layer encryption

- *"gateway-to-gateway"* or *"road warrior"* use cases
- a.k.a. *"Virtual Private Network"* (VPN)

network layer encryption

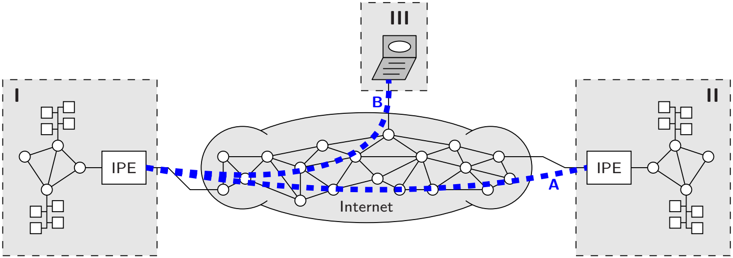


Figure 21: network layer encryption use cases

network layer encryption

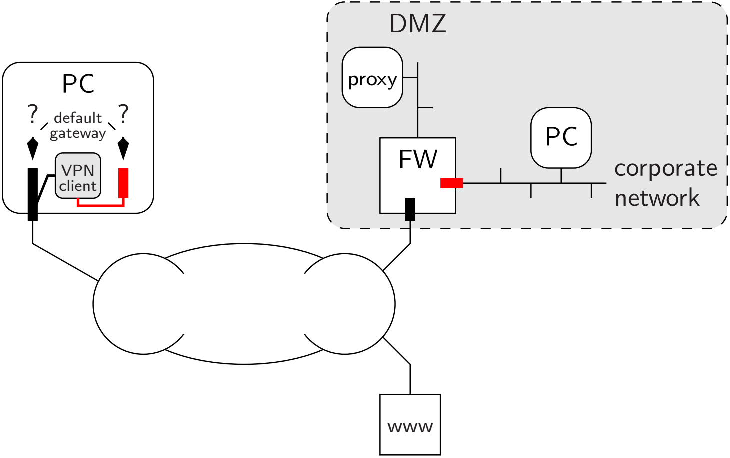


Figure 22: allow *"split tunneling"* or not?

network layer encryption

FBI accused of planting backdoor in OpenBSD IPSEC stack

A former OpenBSD contributor claims that the FBI paid open source developers …

by **Ryan Paul** - Dec 15, 2010 3:11pm CET

113

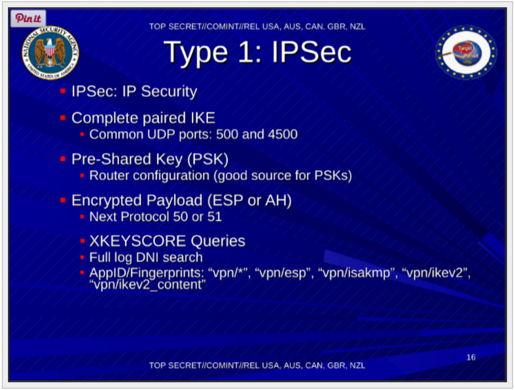
In an e-mail sent to BSD project leader Theo de Raadt, former NETSEC CTO Gregory Perry has claimed that NETSEC developers helped the FBI plant "a number of backdoors" in the OpenBSD cryptographic framework approximately a decade ago.

Perry says that his nondisclosure agreement with the FBI has expired, allowing him to finally bring the issue to the attention of OpenBSD developers. Perry also suggests that knowledge of the FBI's backdoors played a role in DARPA's decision to withdraw millions of dollars of grant funding from OpenBSD in 2003.

"I wanted to make you aware of the fact that the FBI implemented a number of backdoors and side channel key leaking mechanisms into the OCF, for the express purpose of monitoring the site to site VPN encryption system implemented by EOUSA, the parent organization to the FBI," wrote Perry. "This is also probably the reason why you lost your DARPA funding, they more than likely caught wind of the fact that those backdoors were present and didn't want to create any derivative products based upon the same."

Figure 23: backdoors ?

network layer encryption



This is probably the most interesting slide. It basically states that IPsec VPNs are compromised by router compromises that steal the IKE PSKs. This might be a good time to upgrade your router firmware to the latest version, and change to new strong PSKs (or switch it over to RSA 2048+ instead)

Figure 24: backdoors ?

transport layer encryption

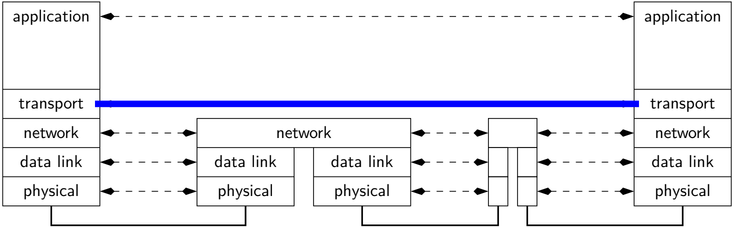


Figure 25: transport layer encryption

- before *"Secure Socket Layer"* (SSL),
  now *"Transport Layer Security"* (TLS)
- the "s" in "https", "imaps", ...

transport layer encryption

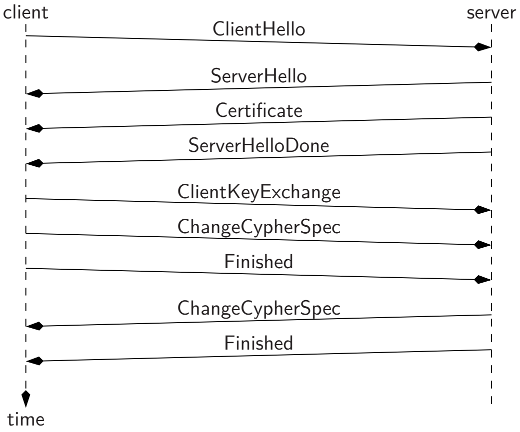


Figure 26: Transport Layer Security (TLS)

- with possibility of *“mutual TLS”* (mTLS)

transport layer encryption

PHONY SSL CERTIFICATES ISSUED FOR GOOGLE, YAHOO, SKYPE, OTHERS

by Paul Roberts

March 23, 2011 , 7:23 pm

UPDATED: A major issuer of secure socket layer (SSL) certificates acknowledged on Wednesday that it had issued 9 fraudulent SSL certificates to seven Web domains, including those for Google.com, Yahoo.com and Skype.com following a security compromise at an affiliate firm. The attack originated from an IP address in Iran, according to a statement from Comodo Inc.

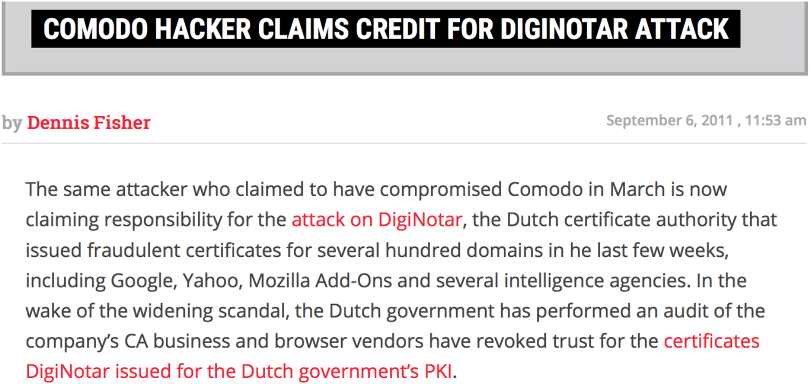
Comodo, of Jersey City, New Jersey, said, in a statement on its Web page, that an attacker was able to obtain the user name and password of a Comodo Registration Authority (RA) based in Southern Europe and issue the fraudulent certificates. The company said the hack did not extend to its root keys or intermediate certificate authorities, but did constitute a serious security incident that warranted attention.

Figure 27: we use certificates so we are safe ?

transport layer encryption

Figure 28: we use certificates so we are safe ?

transport layer encryption



COMODO HACKER CLAIMS CREDIT FOR DIGINOTAR ATTACK

by Dennis Fisher   September 6, 2011 , 11:53 am

The same attacker who claimed to have compromised Comodo in March is now claiming responsibility for the attack on DigiNotar, the Dutch certificate authority that issued fraudulent certificates for several hundred domains in he last few weeks, including Google, Yahoo, Mozilla Add-Ons and several intelligence agencies. In the wake of the widening scandal, the Dutch government has performed an audit of the company's CA business and browser vendors have revoked trust for the certificates DigiNotar issued for the Dutch government's PKI.

Figure 29: we use certificates so we are safe ?

transport layer encryption

RSA BSAFE

From Wikipedia, the free encyclopedia

RSA BSAFE is a FIPS 140-2 validated cryptography library offered by RSA Security. From 2004 to 2013 the default random number generator in the library contained a backdoor from the American National Security Agency, as part of NSA's secret Bullrun program.[citation needed]

Contents [hide]

SSL-C [edit]

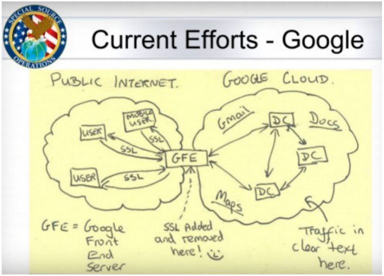
The **SSL-C** library is an SSL toolkit in the BSAFE suite. It was originally written by Eric A. Young and Tim J. Hudson, as a fork of the open library SSLeay, that they developed prior to joining RSA.[1][2] Like SSLeay, SSL-C supported SSLv2, SSLv3, TLSv1; while it also supports X.509v1 and X.509v3.[3] SSL-C was first released in 1999.[4]

Dual_EC_DRBG backdoor [edit]

From 2004 to 2013, the default cryptographically secure pseudorandom number generator (CSPRNG) in BSAFE was Dual_EC_DRBG, which contained a backdoor from NSA, in addition to being a biased and slow CSPRNG.[5] The cryptographic community had been aware that Dual_EC_DRBG was a very poor CSPRNG since shortly after the specification was posted in 2005, and by 2007 it had become apparent that the CSPRNG seemed to be designed to contain a hidden backdoor for NSA, usable only by NSA via a secret key.[6] In 2007 Bruce Schneier described the backdoor as too obvious to trick anyone to use it - there does not seem to have been an awareness that RSA Security had made it the default in BSAFE until the Snowden leak.[6] The backdoor was confirmed in the Snowden leaks in 2013, and it was also revealed that NSA had secretly paid RSA Security $10 million to use Dual_EC_DRBG by default in 2004,[7] though RSA Security denied that they knew about the backdoor in 2004. The Reuters article which revealed the secret $10 million contract to use Dual_EC_DRBG described the deal as "handled by business leaders rather than pure technologists".[7] RSA Security has largely declined to explain their choice to continue using Dual_EC_DRBG even after the defects and potential backdoor were discovered in 2006 and 2007, but has denied knowingly inserting the backdoor.[8]

Figure 30: backdoors ?

transport layer encryption



The NSA intercepts millions of pieces of Google and Yahoo user information each day by tapping into the links between servers, *The Washington Post* reports. According to documents leaked by Edward Snowden, the agency secretly exploits the data links in Google and Yahoo's global networks through a project called MUSCULAR, allegedly operated jointly with the GCHQ (which was accused earlier this year of snagging data from fiber optic cables). A January 9th document says that in the preceding 30 days, collectors had processed over 181 million pieces of information, including both metadata and the actual contents of communications.

Figure 31: backdoors ?

application layer security

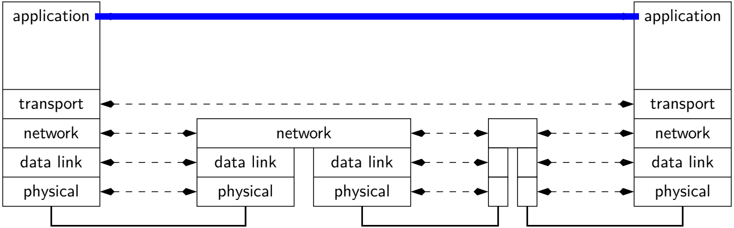


Figure 32: application layer encryption

- when you do not have a direct transport layer connection between the two parties

application layer security

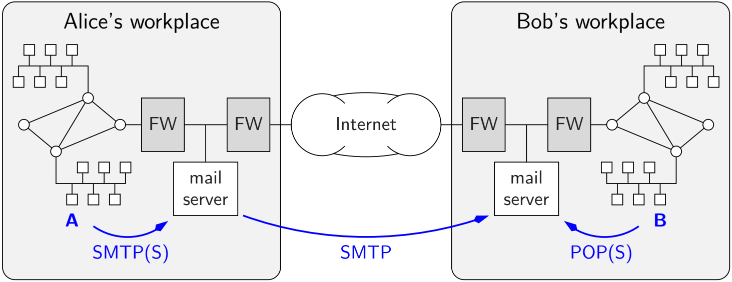


Figure 33: application layer encryption

application layer security

Most Android Apps are Crypto Fails

This study from Carnegie Mellon and UCSB analyzed 11,748 android apps that use crypto and found that 10,327 of them (88%) were flawed. They built a tool to check for *extremely obvious* crypto implementation errors like

- Using ECB mode.
- Using a non-random IV for CBC mode.
- Using constant encryption keys.
- Using constant salts for password hashing.
- Using fewer than 1000 iterations in password hashing.
- Seeding the random number generator with a static value.

Except for the “1000 iterations” one, these are all obvious flaws, and anyone who knows anything about cryptography should know that they are a bad idea. Especially “using constant encryption keys” - that’s *insane*.

Anyway, here are their results summarized in a table.

# apps	violated rule
5,656	Uses ECB (BouncyCastle default) (R1)
3,644	Uses constant symmetric key (R3)
2,000	Uses ECB (Explicit use) (R1)
1,932	Uses constant IV (R2)
1,636	Used iteration count < 1,000 for PBE(R5)
1,629	Seeds SecureRandom with static (R6)
1,574	Uses static salt for PBE (R4)
1,421	No violation

Figure 34: weak crypto

conclusions

conclusions

Figure 35: questions or comments ?