Practical Security

System and Network Administration

Revision 3 (2021/22)

Pierre-Philipp Braun <pbraun@nethence.com>

Table of contents

Security Basics

assuming a server or network device – not a desktop

What are the most important things to do in terms of security?…

==> System & network security in a nutshell

- shutdown unused services + l4 firewall + l3/acl
- system updates
- no passwords (or at least no weak passwords)

How to check what's listening locally again?…

==> that should be clear by now

```
netstat -lntup
```

Also check remotely for clarity and in case a rootkit is hiding some listening port

```
nmap -sTUV -p0-65535 YOUR-SERVER
```

Auto-updates policy

Need to evaluate the risk of downtime

- Debian/Ubuntu restarts the services after upgrade
- but it doesn't for some third-party repositories e.g. nginx.org
- RHEL/CentOS does not
- care about listening daemons mostly
- kernel patch - what part is truly in use? (downtime)

Auto-updates on GNU/Linux

update systems –> licensing lecture

Ubuntu

```
apt install unattended-upgrades
dpkg-reconfigure -plow unattended-upgrades
```

Slackware choice 1:

LAB // PoC autoslack for 14.2 vs current[1] – is it too dangerous to auto-update current and why?

Slackware choice 2: DIY

```
vi /etc/cron.daily/DAILY


# don't do that on current
/usr/sbin/slackpkg update
/usr/sbin/slackpkg -batch=on -default_answer=y upgrade-all
```

[1] http://www.slackware.com/~david/zuul/autoslack/

Auto-updates on BSD systems

Possibly scriptable

- FreeBSD: ?
- DragonFlyBSD: ?
- NetBSD: grab from nightly builds and erase everything but `/etc/`

goes as

```
tar xzpf base.tgz -C /
```

Possibly automated

- OpenBSD: syspatch utility
- MirBSD: *idem*?

Network security in a nutshell

Split activities into VLANs

- e.g. DMZ, VoIP, user, mgmt/backup
- cluster/storage network not routed (but pivot possible)
- eventually ACLs for mgmt

Know your location - what network segment?

- got public IP?
- –or– NAT outbound traffic allowed?
- how many enemies you have?

Isolate insecure industrial tools and software e.g. SCADA

Userland rootkit checkers

some good practice I wouldn't recommend

Regularly check against rootkits

- makes me think of grand-ma who absolutely needs an anti-virus
- it most probably won't detect anything targeted…

LAB // evaluate those detectors against DIY modifications…

mainly GNU/Linux

- Lynis
- Chkrootkit
- Rkhunter
- ClamAV
- LMD

–or– simply overwrite the binaries (BSD & Slackware)

Almost done for this chapter

What were the three most important things to take care of on a server, security-wise?…

==>

- open ports vs. network segments vs. firewall
- KEEP YOUR SERVERS AND NETWORK DEVICES UP-TO-DATE !
- weak passwords = no passwords

// Questions on security basics?

MITM Explained

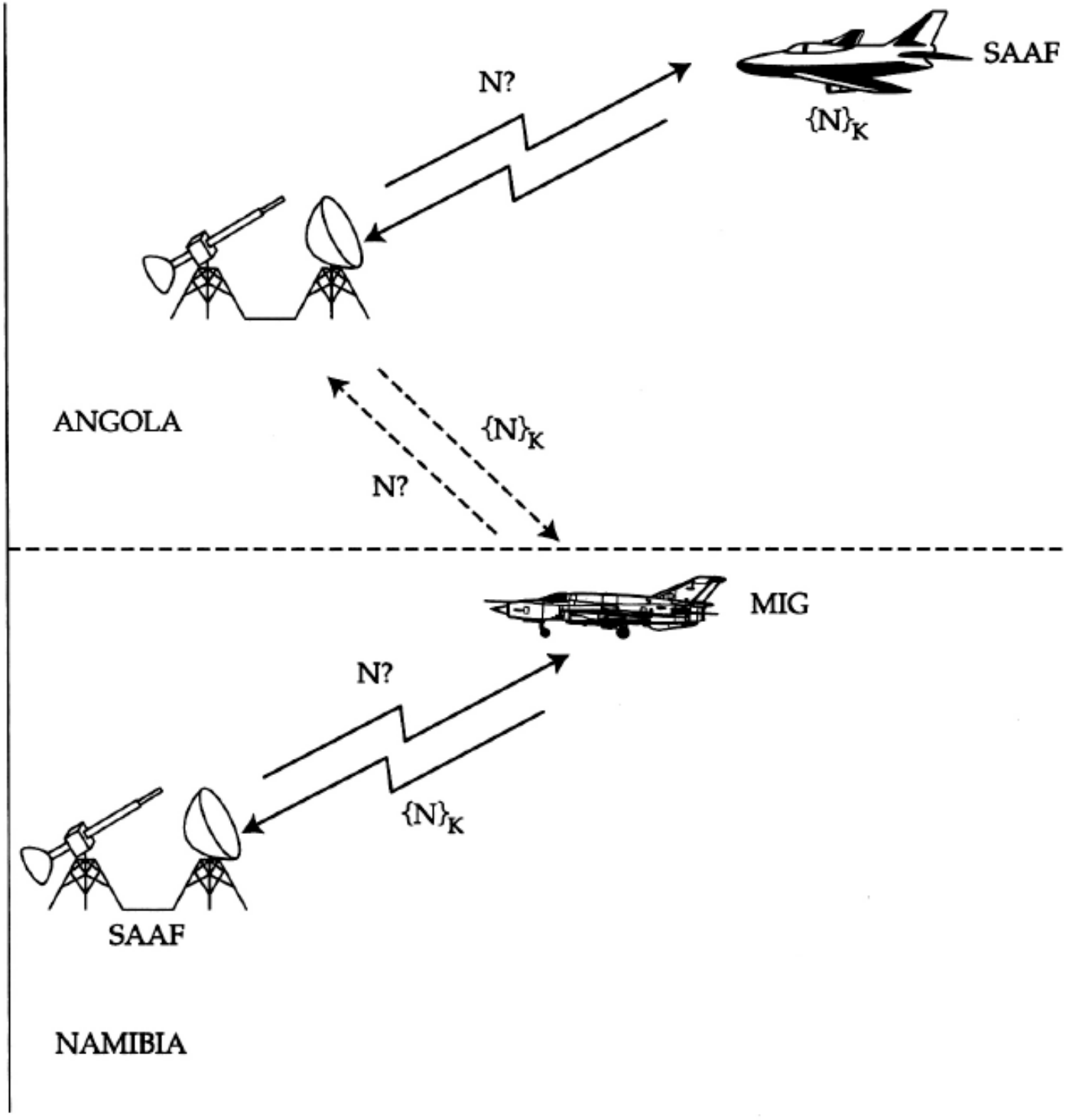
just in case you didn't get it yet

Two ways to explain things

- academic way
- military way

Eve

Mallory


SAAF
N?
{N}K
ANGOLA
{N}K
N?
MIG
N?
{N}K
SAAF
NAMIBIA

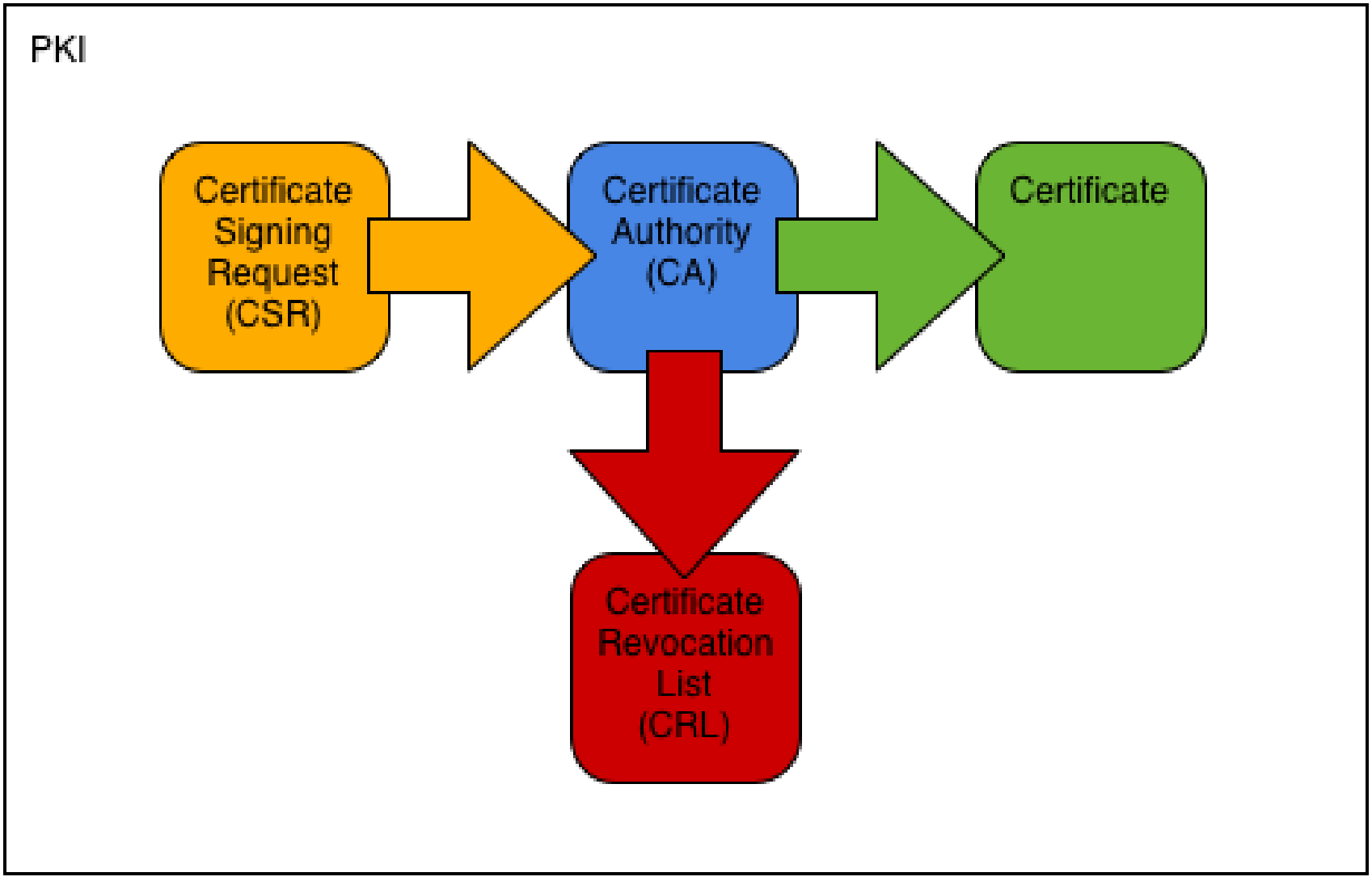
PKIX

What is the purpose of an SSL certificate?…

not just a key pair

==> bind pubkey & domain name **and signed by** (intermediate) **authority**

Looks like there's only one root…



Works many times… **and at the same time**

Which of those three can sign others?…

==> only root CA and intermediates. you just have a leaf-node certificate.

SSL/TLS certificate types

- Signed by official CAs (embedded Mozilla & Chrome)
- Signed by a private CA (pushed to workstations or added once)
- Self-signed (just like a root cert)

Ubuntu ships self-signed for convenience (what about Debian?)

```
/etc/nginx/snippets/snakeoil.conf
/etc/ssl/certs/ssl-cert-snakeoil.pem
/etc/ssl/private/ssl-cert-snakeoil.key
```

// Questions on PKIX?

Applied Cryptography

Categories for crypto

- Symmetric ciphers
- Public Key ciphers
- Hash algorithms & PRNG
- Key negotiation algorithms

*Name a few **symmetric** ciphers (block vs stream)…*

64-bit symmetric block ciphers

DES, 3DES

IDEA

Blowfish (Bruce Schneier)

128-bit symmetric block ciphers

AES (Rijndael, NIST)

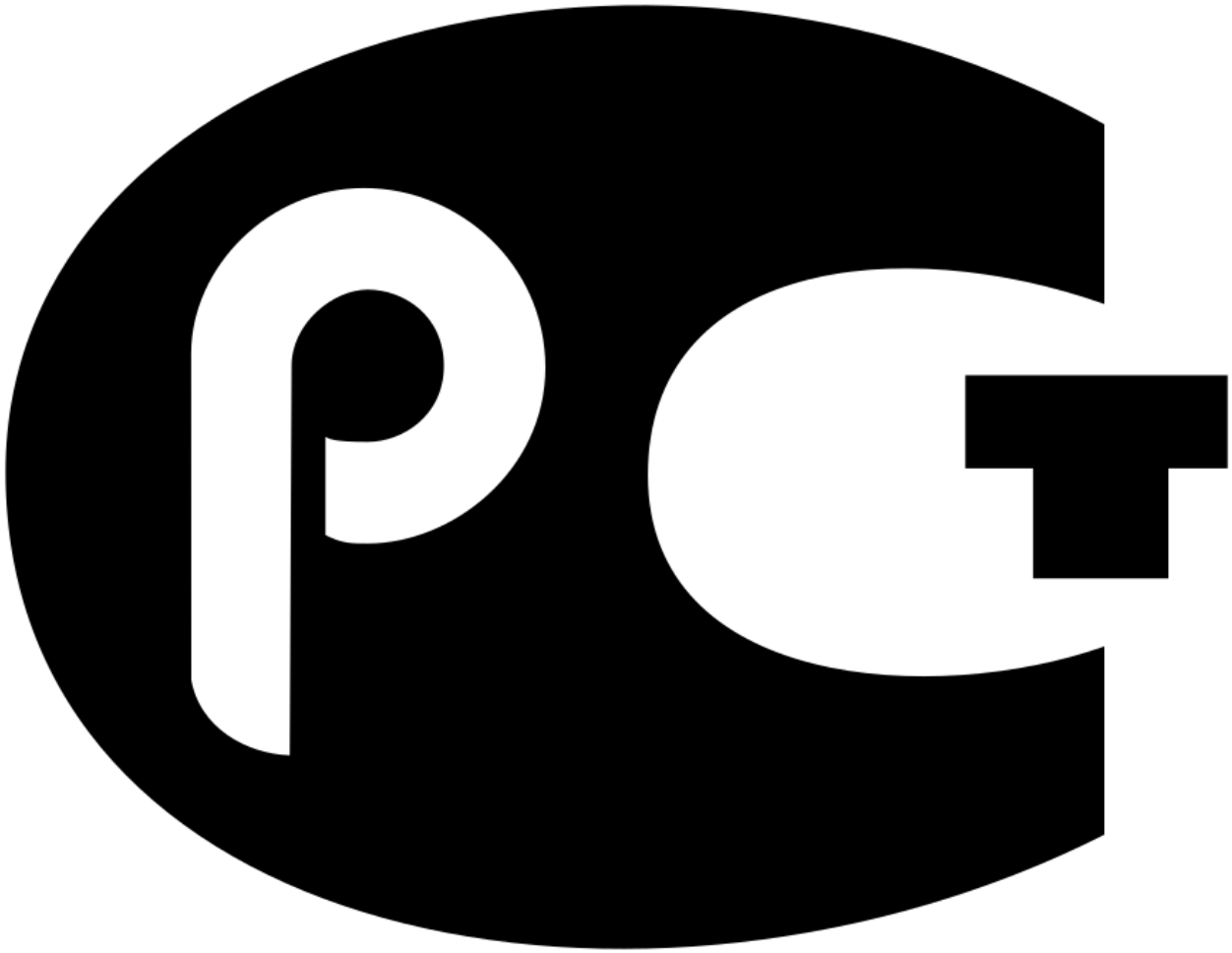
Camellia (Japan)

Twofish (Bruce Schneier)

SEED (South Korea)

ARIA (based on AES, South Korea)

GOST (w/o the H, Russia)



LAB // run ГОСТ encryption between e.g. LibreSSL and GNUTLS

Symmetric stream ciphers

FISH (not seen)

RC4 (deprecated)

CHACHA20

A5/1

Name a few public key ciphers (or signature schemes)…

Public key (asymmetric) ciphers

`RSA encryption`

`ElGamal encryption // LAB`

Signature schemes

`RSA signature`

`DSA`

`ECDSA`

`ElGamal signature // LAB`

`Schnorr signature // LAB`

Typical use-case (SSL)

- RSA/ECDSA to authenticate
- negociates a secret and goes symmetric
- and eventually takes advantage of AES offloading

Modes of operation (for block cipher)

```
ECB (there's a catch)
CBC (to-be deprecated)
CTR (make it a stream)
GCM (idem)
```

What do you do for the last block, if there is not enough data to fit-in?

Padding

For the last block and with non-streaming modes

```
zero-padding (the catch is not that obvious)
PKCS#1 v1.5
PKCS#1 v2.0 + RSAES-OAEP
PKCS#1 v2.1 + RSAES-PSS
```

What about hash functions, any names in mind?…

Hash functions

MD5 (not really deprecated)

SHA-1 (only deprecated for SSL certs)

SHA-2 (SHA-256, ...)

SHA-3 (NIST 2015) -- sponge construction

Note

PKCS#1 v2.2 + SHA 224/256/512

Something about Integrity?…

Loads of acronyms…

- MAC Medium Access Control address (OSI layer 2)
- MAC **Message Authentication Code** (more than just a hash)
- MAC Mandatory Access Control (vs. DAC/RBAC)

- aka protected checksum & error detection code
- aka keyed hash: also message authentication based on symmetric secret
- (sign & verify but using the same secret)
- can be considered as a one-time pad when used for a single message

HMAC — two rounds with inner and outer derived keys

`HMAC-MD5 SHA-1 SHA-2 SHA-3`

Faster MAC with universal hashing

`UMAC x32 optimized`

`VMAC x64+ optimized`

`SipHash (Daniel J. Bernstein)`

`Poly1305 (Daniel J. Bernstein)`

MAC based on mode of operation

`(CBC?) OMAC CCM GCM PMAC`

The special case of AEAD

happy-happy combinations e.g.

```
AEAD_AES_128_GCM
AEAD_AES_128_CCM
AEAD_AES_SIV_CMAC_256
AEAD_AES_128_OCB_TAGLEN64
AEAD_CHACHA20_POLY1305
AEAD_AES_128_GCM_SIV
```

- MAC on both associated data and ciphertext
- MAC depends on context in neighbor messages/blocks

PRNG

- “pseudo”
- cryptographically secure pseudorandom generators (CSPRGs)
- pseudorandom generator theorem –> one-way function

implementations

- stream ciphers (RC4, CHACHA20)
- block cipher with CTR or OFB modes

related to

- trapdoor operation
- hash functions (for the seed only?)

```
/dev/random  -- requires initialized entropy
         -- previously required enough entropy
            (and /dev/arandom behaved like what it now does)
/dev/urandom -- unlimited (non-blocking)
```

```
cat /proc/sys/kernel/random/entropy_avail
cat /proc/sys/kernel/random/poolsize
```

Are there ways to get a better entropy?…

Hardware

HWRNG & rng-tools

Radio-based (using noise)

User-space software

HAVEGE (HArdware Volatile Entropy Gathering and Expansion)

timer_entropyd

randomsound

Name a few key agreement algorithms…

Key exchange algorithms

```
DH
DHE (PFS / ephemeral)
ECDH
ECDHE (PFS / ephemeral)
```

Note it's also possible to simply encrypt the secret and send it to Alice

CIA triad / quadrad / polyad

- Confidentiality
- Integrity
- Availablility
- (Non-repudiation)
- Authenticity (Authentication)
- (Accountability)

Apply secure channels & crypto to those concepts

…which one leverages a secret (symmetric cipher)?

…which one leverages public key cryptography?

…and which one protects against MITM?

==>

- Confidentiality –> symmetric encryption & key agreement (DH)
- Integrity –> hash function (possibly using privkey)
- Availablility
- (Non-repudiation)
- Authentication –> public key crypto
  - –> auth with private key / sign
  - –> and also used for key agreement (RSA)
- (Accountability)

How to authenticate / is any public key or certificate fine?…

==> you need a Trust Anchor

Is it a BI-DIRECTIONAL process?…

==> you need to authenticate both sides if server is not public

Client authenticates server

```
public HTTPS -- SSL -- PKIX chain of trust
```

Stub-client or forwarder authenticates answer

```
DNS -- DNSSEC chain of trust
```

Bi-directional

```
SSH -- client does TOFU fingerprint & PIN host pubkey
SSH -- server checks authorized pubkeys
Wifi -- PSK (when there is)
```

BTS authenticates handset

```
GSM 2G -- SIM card
```

// Questions on practical cryptography?

Tips & Tricks

- initiate an SSL session & read an X.509 certificate
- remote sniffing

Initiate an SSL session (becomes telnet)

```
openssl s_client -connect archlinux.org:443
```

```
Q
```

Read an X.509 certificate

```
openssl x509 -in domain.crt -noout -text
```

Can we do both at once?…

==>

```
echo Q | openssl s_client -connect archlinux.org:443 \
    | openssl x509 -noout -text
```

Super-duper remote sniffing

```
ssh -l root GOT_MIRROR \
"/usr/sbin/tcpdump -n -e -i eth3 -s0 -w - " \
| wireshark -k -i -
```

```
-e also show MAC addreses
-s0 backward compatible w/ now default packet snapshot
    length of 262144 bytes
```

- The interface is plugged to a dedicated port-mirror here
- …otherwise need to filter out ssh itself)

This is the end