Lecture Notes

Esecurity: secure internet & e-voting

Michael Nüsken

b-it

(Bonn-Aachen International Center
for Information Technology)

Summer 2013

```
Return-Path: <auto-notification@fiserv.com>
Received: from postfix.iai.uni-bonn.de (uran.iai.uni-bonn.de [131.220.8.1])
     by mailbox.iai.uni-bonn.de with LMTPA;
     Tue, 09 Apr 2013 00:14:07 +0200
X-Sieve: CMU Sieve 2.4
X-IAI-Env-From: <auto-notification@fiserv.com> : [131.220.8.23]
Received: from mandos.iai.uni-bonn.de (mandos.iai.uni-bonn.de [131.220.8.23])
    by postfix.iai.uni-bonn.de (Postfix) with ESMTP
    id 0FCBB5C40A; Tue,  9 Apr 2013 00:14:07 +0200 (MEST)
    (envelope-from auto-notification@fiserv.com)
    (envelope-to VARIOUS) (2)
    (internal use: ta=0, tu=1, te=0, am=-, au=-)
X-IAI-Env-From: <auto-notification@fiserv.com> : [127.0.0.1]
Received: from localhost (localhost [127.0.0.1])
    by mandos.iai.uni-bonn.de (Postfix) with ESMTP id BF050484;
    Tue,  9 Apr 2013 00:14:06 +0200 (MEST)
    (envelope-from auto-notification@fiserv.com)
    (envelope-to VARIOUS) (2)
X-Amavis-Alert: BANNED, message contains part: multipart/mixed |
    application/zip,.zip,PaymentAdvice.zip |
    .exe,.exe-ms,Payment_Advice.exe
Received: from mandos.iai.uni-bonn.de ([127.0.0.1])
    by localhost (mandos.iai.uni-bonn.de [127.0.0.1]) (amavisd-new, port 10024)
    with ESMTP id hnNY-skV3fIz; Tue,  9 Apr 2013 00:13:57 +0200 (MEST)
X-IAI-Env-From: <auto-notification@fiserv.com> : [75.146.222.245]
Received: from mail.westwindsorpolice.com (mail.westwindsorpolice.com [75.146.222
    by mandos.iai.uni-bonn.de (Postfix) with ESMTP id 5F7BA7E;
    Tue,  9 Apr 2013 00:13:52 +0200 (MEST)
    (envelope-from auto-notification@fiserv.com)
    (envelope-to VARIOUS) (2)
Received: from [142.113.244.215] (port=92486 helo=[192.168.8.85])
    by 75.146.222.245 with asmtp
    id 1rqLaL-000WF-00
    for [redacted]@bit.uni-bonn.de; Mon, 8 Apr 2013 17:13:54 -0500
Message-ID: <51633D34.6040909@fiserv.com>
Date: Mon, 8 Apr 2013 17:13:54 -0500
From: Gertrude_Childers@fiserv.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:7.0.1) Gecko/20110929 Thunderb
MIME-Version: 1.0
To: [redacted]@bit.uni-bonn.de
Subject: Payment Advice - Advice Ref:[B3950667]
Content-Type: multipart/mixed;
 boundary="----=_Part_95880_5736334738.9742644781979"
X-Spam: Not detected
X-Mras: Ok
X-Spam-Level: ********************** at mandos.iai.uni-bonn.de
X-Spam-Score: 23.1 at mandos.iai.uni-bonn.de
X-Spam-Status: No, score=23.134 tagged_above=9999.9 required=9999.9
    tests=[AV:Sanesecurity.Junk.44718.UNOFFICIAL=0.1,  COPYRIGHT=0.1,
    DEAR_SOMETHING=2.234, IAI_10288a12=0, IAI_10323a14=0, IAI_10324a15=0,
    IAI_10330a14=0, IAI_10336a15=0, IAI_10337a14=0, IAI_10360a14=0,
    IXHASH=5.5, L_AV_SS_Spam=8, RCVD_IN_XBL=4.4, SPF_FAIL=2.8]
X-Spam-Report: AV scanner ClamAV-clamd-ws reported spam (not infection):
    Sanesecurity.Junk.44718.UNOFFICIAL
    namo@mandos.iai.uni-bonn.de pronounced judgment in matters of
    spam: Final score: 23.0 points
    pts rule ------------description----------------------------------------
    8.0 L_AV_SS_Spam           L_AV_SS_Spam
    4.4 RCVD_IN_XBL            RBL: Received via a relay in Spamhaus XBL
                               [75.146.222.245 listed in zen.spamhaus.org]
    2.8 SPF_FAIL               SPF: sender does not match SPF record (fail)
    [SPF failed: Please see http://www.openspf.org/Why?s=mfrom&id=auto-notificati
    2.2 DEAR_SOMETHING         BODY: Contains 'Dear (something)'
    0.1 COPYRIGHT              BODY: COPYRIGHT
    5.5 IXHASH                 BODY: iXhash manitu.net says its spam
    0.0 IAI_10337a14           IAI_10337a14
    0.0 IAI_10323a14           IAI_10323a14
    0.0 IAI_10288a12           IAI_10288a12
    0.0 IAI_10330a14           IAI_10330a14
```

```
    0.0 IAI_10336a15           IAI_10336a15
    0.0 IAI_10324a15           IAI_10324a15
    0.0 IAI_10360a14           IAI_10360a14
    If you have any questions, see https://mailbox.iai.uni-bonn.de/anti.html
X-Virus-Scanned: amavisd-new (Kater5) at mandos.iai.uni-bonn.de

This is a multi-part message in MIME format.
------=_Part_95880_5736334738.9742644781979
Content-Type: text/plain; charset=windows-1251; format=flowed
Content-Transfer-Encoding: 7bit

Dear Sir/Madam

Upon your request, attached please find payment e-Advice for your reference.

Yours faithfully

HSBC

**********************************************************************

We maintain strict security standards and procedures to prevent unauthorised acce

Please do not reply to this e-mail. Should you wish to contact us, please send yo

Note: it is important that you do not provide your account or credit card numbers

Copyright. The HongKong and Shanghai Banking Corporation Limited 2013 All rights

**********************************************************************


------=_Part_95880_5736334738.9742644781979
Content-Type: application/zip;
 name="PaymentAdvice.zip"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
 name="PaymentAdvice.zip"

UEsDBBQAAAAIAAqFiELY5e54oIwBAAD+AQASAAAAUGF5bWVudF9BZHZpY2UuZXhl7FsJeFRV
sj5ZIJERgZlRZ97gGDTfe27DRPbxOU4iy4AG7CykIy5JkzRNIHSwuwPBpw4QXPKYAOM6ggvv
ocLouLDIIPJkkUURBGNkSUIgCUmaJCS9b3epqTrnJmkSiEvP+763cLBy7q1zquqvOnXq3HsT
p85YxWIYY7FIAIxtY6Ils75bFNJipI+HsKuu33LF4WHbolIPD1u1KnN2gTVhvqXIZDHMS5hX
bLUlzDQmWIrNCcXmfKMlQV9gHjliwMDEsZ16dBMZS42KZS//M0vr5J1hg2J+FBXP2JV40x7F
eQ2D8QdRAhPo6DqasX5MjHf2LClaOMOHo4TQYBbWd3W87UB9Ok3lqMao7gHkb2D/fW24zVhi
w/7EAA3QlawLdxiE3OGWfIPNwFhFrOZ7/x4OCOjJw8U0RupYkogNu6bXvF3DLVZLHtN8pUVu
wv5nveexy+3/RVsVYcuevryttG2UHsbC2PR0++2YpqV7B+d8MiEhmS0fkwULh86AiuWND32K
c0fuyl/JsnPtnwEAF8vmYvZnuFB8Dk3j/ETBL4nuwc/W69KRfyXnX9nNz9VnpKXhgD9OWNfs
DVr672jp05Gfl28cOiyZle5/bvkkWH50957Bs0ZW2Meg9/lwe37ZsCUtCVj5Ft+hs8WU7okv
X/pXHEkZ`PXHtxxcmLDxF7fELxyyKXnc4IU/Kv8tS0pmj23IhqOvzomD0YvRx08ZM/uGs2gN
B8edekUP3Hfqs3RZ9sNhsFdxfibOz8hAiQEsHDgfSxa6no/r1tX3SoTJZdmjwkIn+JP1uow0
+7v9e2JIovnc1KL4MBB8bJwYsxexMKnv37p0cTNmXJNuZVocaPXq+vfEPEqfzkWk2D7t87kP
6nkC2tRw7X03Lde4DZ19QFyY+906dWj/2pjv5b8WV0rVK1hPn3RiXWN75vZkfUZGut3bK0cm
62kFivt1T+/Df/vo2J72sjvjfnf/vpzgc1P1CGF/TJgKwU/Qp/Pgvh99wf4Ky/kse9wFy9oZ
W4weheGeCwUvZnuoPoMcKIzqsbeTtByYq3z3he3USWWJMmuh3Btbpp52XpbdFNXbp1Tal+n2
cXCB2MX2WjqvO//Sr7eOUVrcdfbP+3Jeywkdr5/2t7jFC3IwVZ+ewQNwS++x20Q9XBz9w7en
liNp6falIdIf39ca3aZPo2qVLHXHpSteWpZt6cPZrnwRK3qdEuZQV65iPDPs5vChXu0Cm/aZ
0FdizGFRtyVmPoTrMYdllo+mJ56s8oElzsN/TFsSm1hy9Qhr+XVsVDLLgNG3YTm/f0prNNTZ
g1xn7LYhs+5o/9WQEaHzpW2xZbGJ2WW/H7oc/zuxLcdXXatbMipxyI2ra7NvrVhycNi7E8tH
fr48SafPyMnKficnu/TWqNt/AknlybHlD8TOGHSwdHfskqAD92Bx1SVjk437j/t0sPd5PMvr
wOE+F+liOhP1GSK3VsO3xpRqdFqaPV7+lrzPpTrxlBQORbOVJur1RfbbZL5eGfaO4Pffx+P0
ujT7JPlb8/NBUVsT+wiSlseIJMue3bPWUG7qMuwbevIn0/60L+lVywX/t997TegM0Nlv9vch
x+fdxM/yKRA2rws/Gmb+sIF/cOusb+n2Bb3XWdQ2+w7/pe13PVtk2EeHeuKfoE/LSrf/PHzj
hp0lOvtp78XyR+RWjNR7LBvrsF31haNhl9t3aoNvZGzFDYy5kLLxegySZxhj9Xj/HtJNSAuQ
Nw77x5AGIO1A+k+kCqRZOHYDEkOqQbn5yJuE1z/DfhrSjBuEnbXIO4R0RyJjVyExpLfx/jbs
n8V+B1ItUhuSD+mfEoWcGfvnkWYgLdV4W7U+YVi3H8tuFO/Hz4TxpuG82cjbFcZ7G31rkVcZ
xtuLvC0J7P9cfO6ISTUaFhjHWwpsBXmGwgxjnq2gyMyYPer3RtukgkJjis1mKZhZbDNa9Ywt
Z+MtRoPNSAN69nFsRrF1vtGcnzkbuflsYMwUa5ahsCA/tQhVGdkB0jG+2GIxmm1iypR89Khf
odVmKTSaUxhrj55ithkthUV5c435E0vyZhvMJiOLYyn5+Sm2onlo0RmbVWCxFRsKdZYiG4JD
```

```
Return-Path: <xelup@volia.net>
Received: from postfix.iai.uni-bonn.de (uran.iai.uni-bonn.de [131.220.8.1])
     by mailbox.iai.uni-bonn.de with LMTPA;
     Sun, 07 Apr 2013 15:59:22 +0200
X-Sieve: CMU Sieve 2.4
X-IAI-Env-From: <xelup@volia.net> : [131.220.8.23]
Received: from mandos.iai.uni-bonn.de (mandos.iai.uni-bonn.de [131.220.8.23])
    by postfix.iai.uni-bonn.de (Postfix) with ESMTP
    id 1BFEE5C401; Sun,  7 Apr 2013 15:59:22 +0200 (MEST)
    (envelope-from xelup@volia.net)
    (envelope-to VARIOUS) (4)
    (internal use: ta=0, tu=1, te=0, am=-, au=-)
X-IAI-Env-From: <xelup@volia.net> : [127.0.0.1]
Received: from localhost (localhost [127.0.0.1])
    by mandos.iai.uni-bonn.de (Postfix) with ESMTP id F363C83;
    Sun,  7 Apr 2013 15:59:21 +0200 (MEST)
    (envelope-from xelup@volia.net)
    (envelope-to VARIOUS) (4)
Received: from mandos.iai.uni-bonn.de ([127.0.0.1])
    by localhost (mandos.iai.uni-bonn.de [127.0.0.1]) (amavisd-new, port 100
    with ESMTP id or7arX53hEMH; Sun,  7 Apr 2013 15:59:13 +0200 (MEST)
X-IAI-Env-From: <xelup@volia.net> : [93.73.226.128]
Received: from crowning.thinker.volia.net (crowning.thinker.volia.net [93.73.
    by mandos.iai.uni-bonn.de (Postfix) with SMTP id 43F187E;
    Sun,  7 Apr 2013 15:58:37 +0200 (MEST)
    (envelope-from xelup@volia.net)
    (envelope-to VARIOUS) (4)
Message-ID: <420047834585.35283771216380@volia.net>
Date: Sun, 07 Apr 2013 15:58:14 +0200
From: Ruby Support <xelup@volia.net>
To: <goenna@bit.uni-bonn.de>
Subject: =?iso-8859-1?B?U2llIHdlcmRlbiBhdXNnZXfkaGx0LCBlaW5lbiBlaW56aWdhcnRp?
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable
X-Spam-Level: ********************* at mandos.iai.uni-bonn.de
X-Spam-Score: 20.9 at mandos.iai.uni-bonn.de
X-Spam-Status: No, score=20.857 tagged_above=9999.9 required=9999.9
    tests=[IAI_00008=0.1, IAI_00011=0.2, IAI_10025=0.1, IAI_10266h12=0.1,
    IAI_10331a12=0, IAI_10347a15=0, IXHASH=5.5, MIME_8BIT_HEADER=0.3,
    RCVD_IN_PBL=3.3, RCVD_IN_XBL=4.4, URIBL_BLACK=4, URIBL_JP_SURBL=2.857]
X-Spam-Report: namo@mandos.iai.uni-bonn.de pronounced judgment in matters of
    spam: Final score: 20.9 points
    pts rule ------------description----------------------------------------
    3.3 RCVD_IN_PBL            RBL: Received via a relay in Spamhaus PBL
                               [93.73.226.128 listed in zen.spamhaus.org]
    4.4 RCVD_IN_XBL            RBL: Received via a relay in Spamhaus XBL
    4.0 URIBL_BLACK            Contains an URL listed in the URIBL blacklist
                               [URIs: jarttoprubygrand.com]
    2.9 URIBL_JP_SURBL         Contains an URL listed in the JP SURBL blockli
                               [URIs: jarttoprubygrand.com]
    0.1 IAI_10025              BODY: IAI_10025
    0.1 IAI_00008              BODY: IAI_00008
    0.2 IAI_00011              BODY: IAI_00011
    5.5 IXHASH                 BODY: iXhash manitu.net says its spam
    0.3 MIME_8BIT_HEADER       Message header contains 8-bit character
    0.1 IAI_10266h12           IAI_10266h12
    0.0 IAI_10347a15           IAI_10347a15
    0.0 IAI_10331a12           IAI_10331a12
    If you have any questions, see https://mailbox.iai.uni-bonn.de/anti.html
X-Virus-Scanned: amavisd-new (Kater5) at mandos.iai.uni-bonn.de
```

DNS

Thunderbird: Ctrl+U

Body

9.4.13
es
4

Email

First email ≈ 1971

SMTP originating before the Internet (1982)

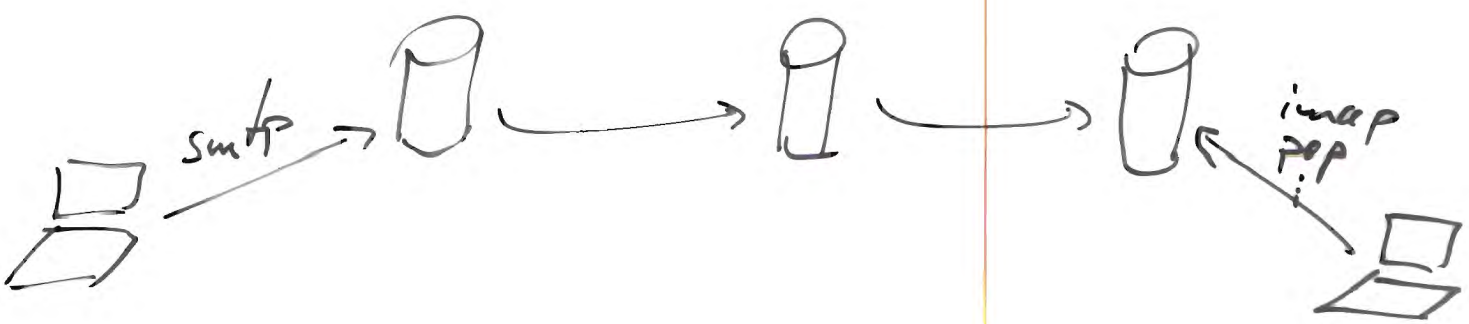
Goal

- Send message
- fast
- to the ~~correct~~ intended receiver
- from any sender to any receiver, anywhere.
- easy-to-use.
- light weight, simple.
- direct

Format

- pure text, electronic, based on ASCII
- simple formats

  <header> : <keyword> : <info>
  <blank line>
  <body>



10.4.13
es
(1)

Security?

→ What do we have?
→ What do we want?
→ History and design?

What is email? → see yesterday.

Bases: SMTP Simple Mail Transfer Protocol
DNS Domain Name Service

What security do we want?

- nobody apart from the intended receiver should be able to "see" the content
  confidentiality
  ↳ encrypt
- be sure that the "sender" is the sender.
- that the message is not modified during transport.
  authentication
  ↳ sign
  integrity
  ↳ sign
- identify recipient, i.e. make sure that the used keys belong to the desired entity.
- identify senders
- reliability → message reaches receiver ... (DoS)
  → service is available at all times

Safety

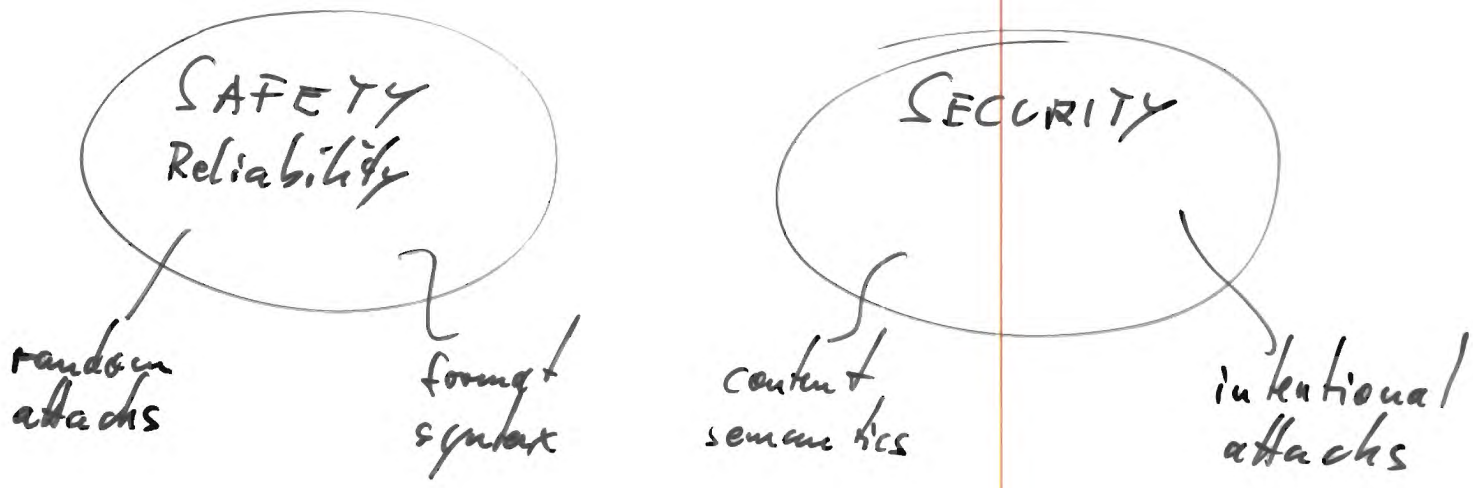
- sender cannot deny what he has written
- proof of submission
  of delivery
  of reading

non repudiation
↳ sign
...

- anonymity of the sender
  ... but still prove authentication

anonymity

The list of requirements splits in two parts:



10.4.13
es
(3)

Attack (vs Email via)

- Intercept emails & reads them
- Modify / forge emails
- Denial of Service (DoS)
- Spoofing (pretend a different sender...)
- SPAM
- Phishing, social engineering
- Send malware

Defense

Encryption
Signature
?

CRYPTO → GnuPG (PGP)

... filtering & security policy

Training, education.

... anti viruses
... security policies
... senders trust + auth.

Outlook

- Toolbox : black boxing
- Internet relevant security protocols
  - IPsec
  + TLS/SSL + SSH

e-voting ...

16.4.15
es
G

For email we may use

GnuPG / PGP

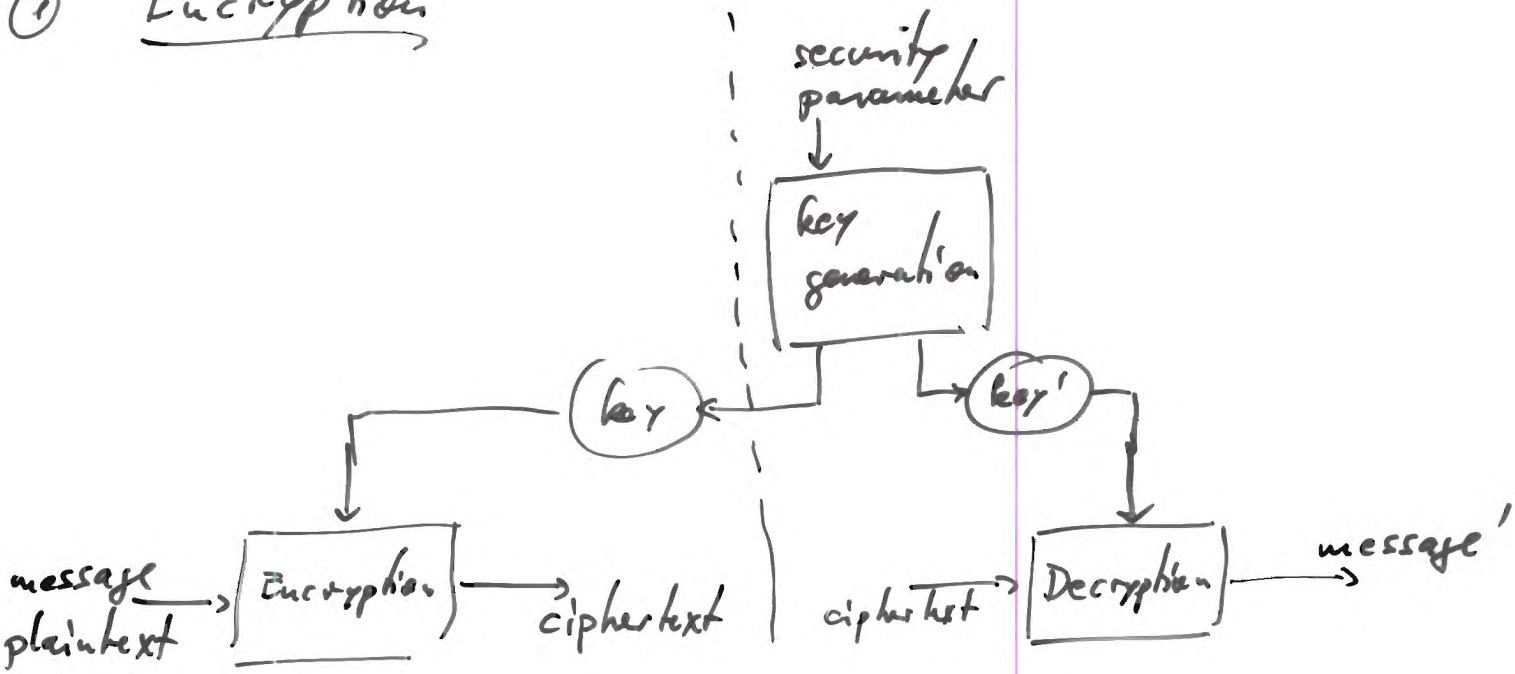
- PGP was developped by Phil Zimmerman in USA.
- Not exportable: >40 bit crypto was considered a military weapon.
  But books did not fall under this regulation.
  Later PGP was sold and became closed source but that didn't make profit and so it was sold again
- However, in the meantime GnuPG was started, again open source and based on the last free version of PGP...

→ More details on Wikipedia...

16.4.13
es
(2)

Toolbox: cryptography & related

(1) Encryption



Conditions:

CORRECTNESS: message' = message ...

EFFICIENCY:
- polynomial time,
- fast (seconds for reasonable security parameters)

SECURITY: ???

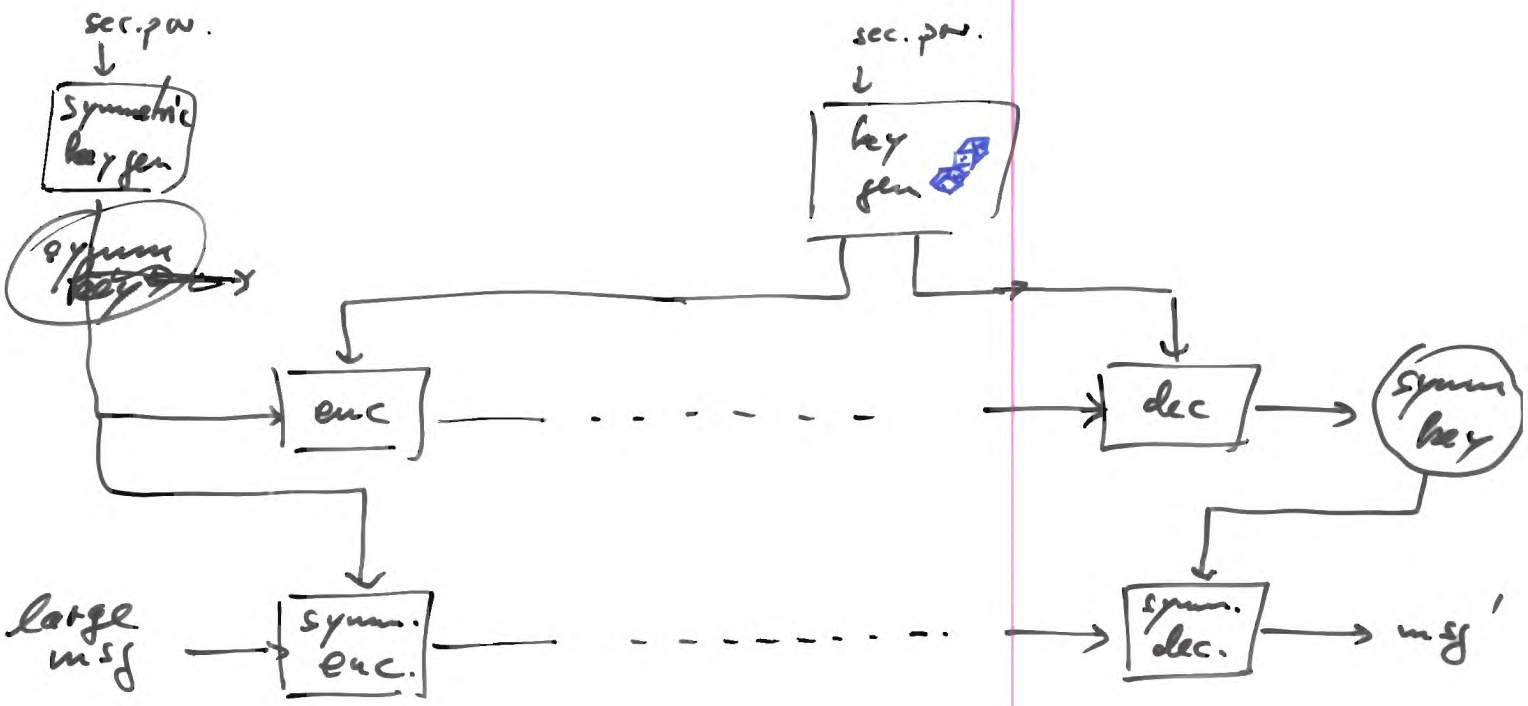
symmetric case: key = key'.

public-key case: key, key' are somehow related but such that it is difficult to derive key' from key

Pro/Cons for public key

- \- slower
- \+ need no prior keys, can use insecure channel for transmitting the public key.

Hybrid encryption combines both worlds and provides all pros!



- Encryption protects against disclosure of the plain text and grants confidentiality.

! No protection against changes!

Kerckhoffs' principle

! The only thing unknown to the attacker is the key.

... or rather the random seed. [entropy ← only 16.5 bit] Debian bug

Repeat:

We need good randomness.
This is the only thing an attacker cannot predict.

Secrets are what remains unpredictable.

Debian bug: entropy ≈ 16.5 bits.

Large internet computations accumulate 2^{60}, maybe 2^{70} operations.

Need ≥ 80 bits of entropy, unpredictable bits.

Remaining open question:

What do we consider as SECURITY ?

Def (Diana)
The scheme is secure if it takes roughly exponential time in the length of the ciphertext to find the plaintext without knowledge the private or secret key.

Def (Mustafizur)

Brute force, i.e. trying all keys, should be the only way to break the scheme, i.e. to find the plaintext.

Security notions

Tasks for the attacker

UBK Unbreakability:
find private or secret key.

⋮

IND Indistinguishability
distinguish the encryptions of two attacker-chosen messages.

NM Non-malleability
change a given ciphertext and tell a (meaningful) rule how the plaintext changed.

Means of the attacker

Always: restricted to some time bound,
eg. polynomial-time,
with a non-negligible advantage.
↗ ie. more than inverse polynomial ↖
} 1 prob of success
− prob of guessing }

The attacker also always gets
all public information.
What else:

KO key only.
Nothing else.

CPA chosen plaintext attack.
Attacker may ask for encryptions
of any plaintext he chooses.
This is only interesting in the symmetric setting

CCA chosen ciphertext attack. [CCA$_2$, ACCA]
Attacker may ask for decryptions
of any ciphertext he chooses.

The attacker wins if

(1) he fulfills the task.

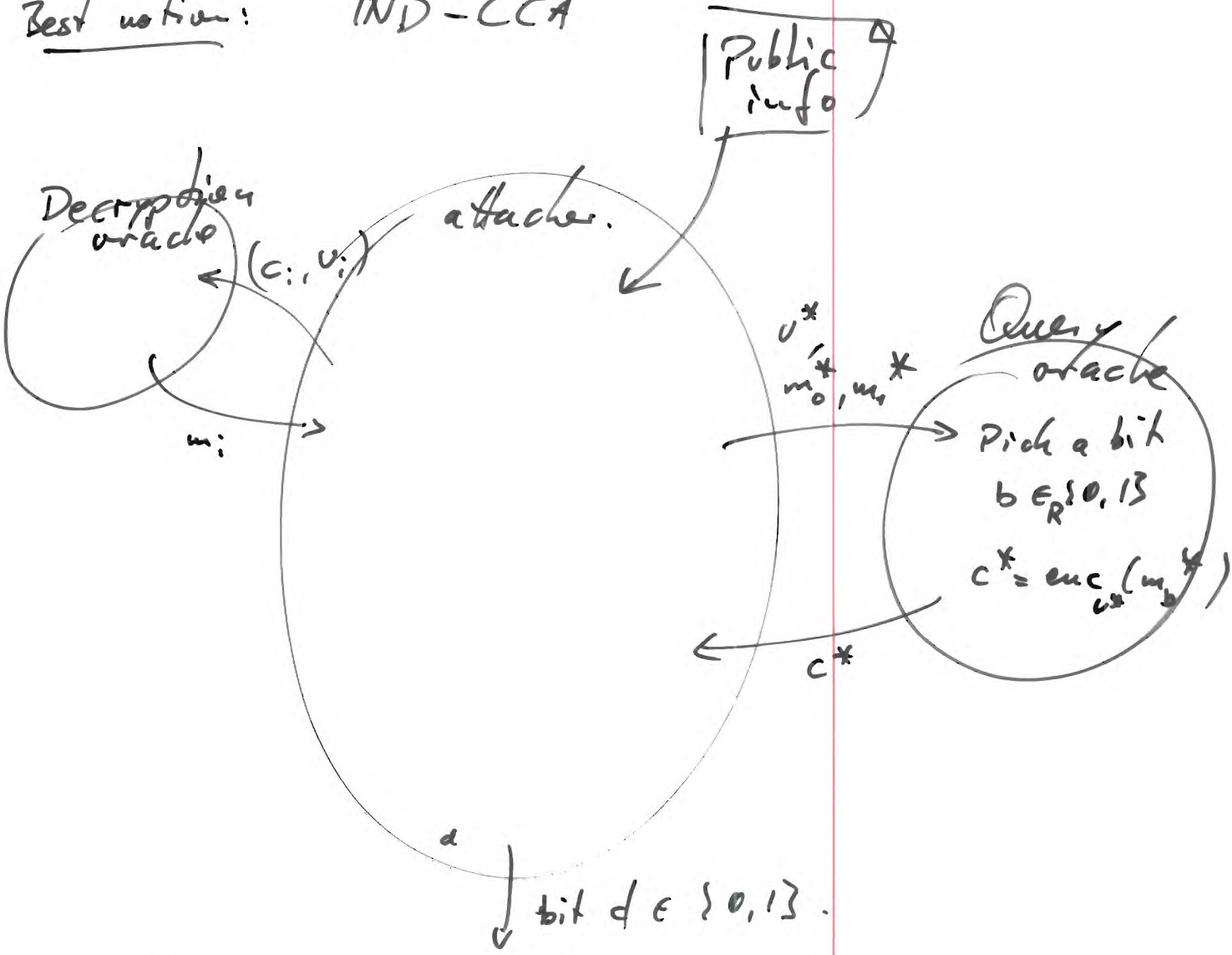
(2) within the bounds of his means and using the additional "oracles".

(3) he doesn't "overuse" his oracles,

ie. in our context, eg.

he does never ask the ciphertext that he has to distinguish to the oracle.

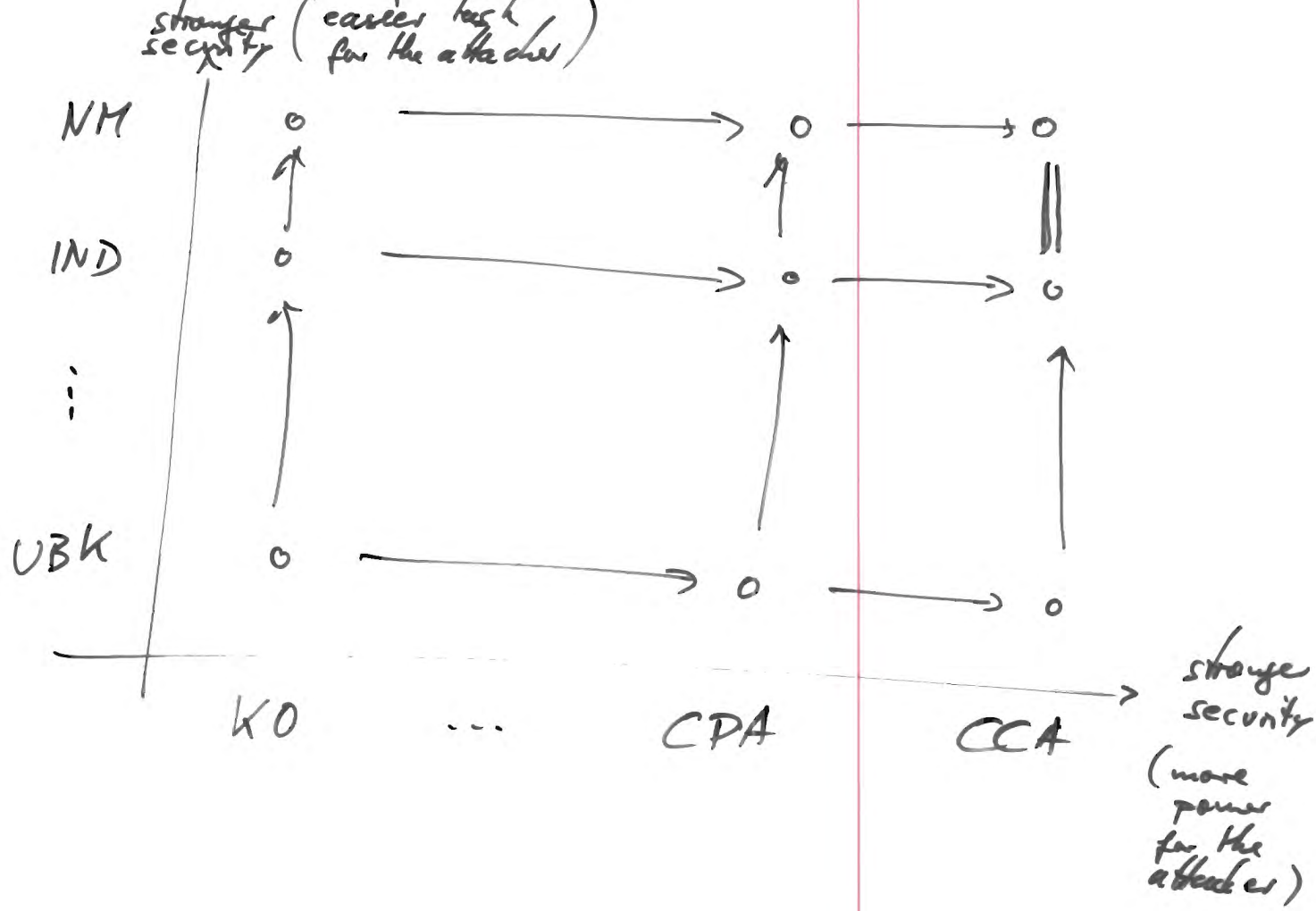
Best notion: IND-CCA



Attacker wins if $b = d$ and (c^*, v^*) was never asked to the decryption oracle.

The encryption scheme is IND-CCA secure if no attacker running in polynomial time with non-negligible ~~success~~ advantage exists.

Similarly, we may combine attack goal and means.

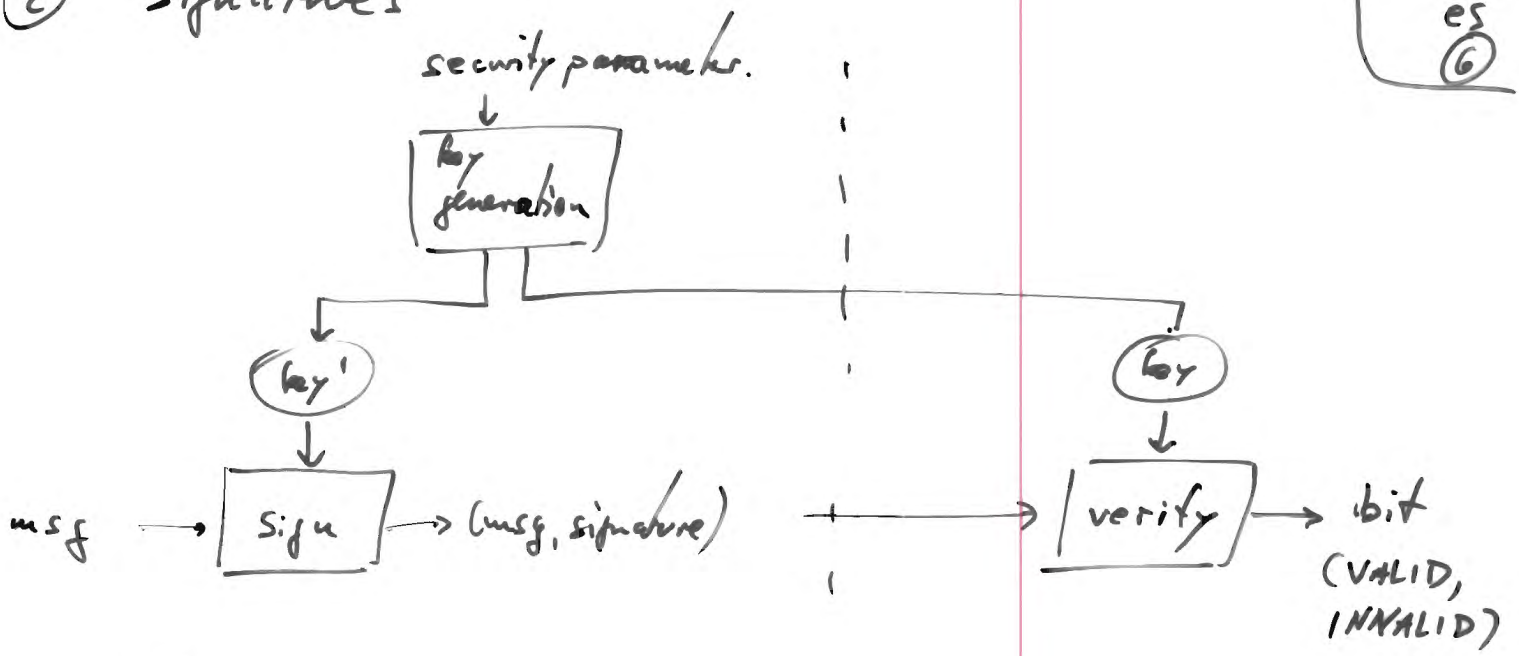


Theorem RSA encryption is not IND-CCA secure, even not IND-KO secure.

The same is true for any deterministic encryption scheme.

We hope that RSA is UBK-CCA secure.

(2) Signatures



Conditions:

CORRECTNESS: bit = VALID, ie.

verify (key, sign (key', msg)) = VALID

⇑

(key, key') = keygen (secpar)

EFFICIENCY: all blocks are "fast".

SECURITY: ???

symmetric case: (key) = (key')

public-key case: key, key' are somehow related but such that it is difficult to derive (key') from (key).

In symmetric case we call "signature" a message authentication code or an integrity check value.

Comparison of public-key and symmetric variant

23.4.13
es
(7)

public-key version:

- integrity: msg is unchanged between sign and verify.
- authenticity: msg originates from an entity associated with the used public key.
- non-repudiation: the signer cannot deny having signed the message.

symmetric version:

- integrity: msg is unchanged between sign and verify by a third party.
- authenticity: msg originates from one of the entity knowing the secret key.

  ⇝ We have no non-repudiation!

Bottom line: symmetric features are (necessarily) weaker!

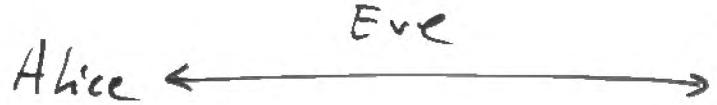
Further tools:

(A) How to associate a public key and a person or entity?

(B) How to exchange a secret key?

How to agree on a shared secret?

Want:



Such that $s_A = s_B$ and Eve cannot derive this from the exchanged messages.

The answer is the Diffie-Hellman key exchange.

First, fix a group G, for example $G = (\mathbb{Z}_p^\times, \cdot)$ for some prime p (multiplicate group of the field $\mathbb{Z}_p = \mathbb{F}_p$, i.e. $\mathbb{Z}_p^\times = \mathbb{Z}_p \setminus \{0\}$; operation is multiplication)

or $G = (E(y^2 = x^3 + ax + b), +)$ for $a, b \in \mathbb{F}_q$.

Def: Properly defined
Associative
Neutral element
Inverse (number)

23.4.13
ec
(3)

It turns out that

$$\mathbb{Z}_{p-1} \underset{\text{dlog}}{\overset{\exp_g}{\rightleftarrows}} \mathbb{Z}_p^\times \qquad , p-1 = \#\mathbb{Z}_p^\times$$

$$a \longmapsto g^a \qquad \vdots \; \ell$$

is often seemingly one-way.

where we assume that $g \in \mathbb{Z}_p^\times$ is a generator,

ie. the image of $\exp_g$ is everything.

$\exp_g$ can be computed suitably fast.

Similarly

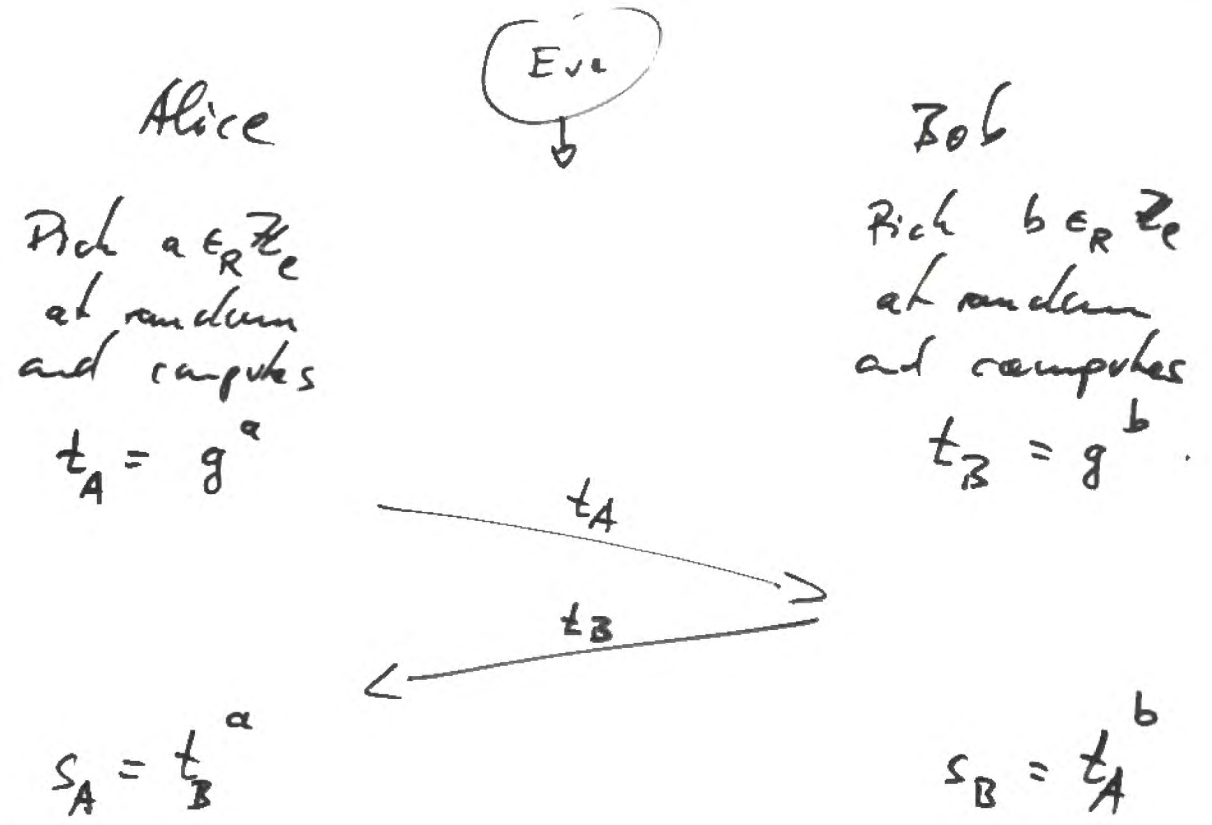
$$\mathbb{Z}_e \underset{\text{dlog}}{\overset{\text{scalar mult}}{\rightleftarrows}} E \qquad , e = \#E,$$

$$a \longmapsto aP$$

for some generator $P \in E$ is easy to compute and often seemingly one-way.

Assume that the discrete log problem for $(G, \cdot)$ with a chosen generator g is hard to solve.

Now, DH does this:



Now: $s_A = t_B^a = (g^b)^a = g^{b \cdot a}$

$= g^{a \cdot b} = (g^a)^b = t_A^b = s_B$.

Eve knows: $(G, \cdot)$, g, $t_A = g^a$, $t_B = g^b$ } DHP (Diffie-Hellman-Problem)

wants: $g^{a \cdot b} = s_A = s_B$.

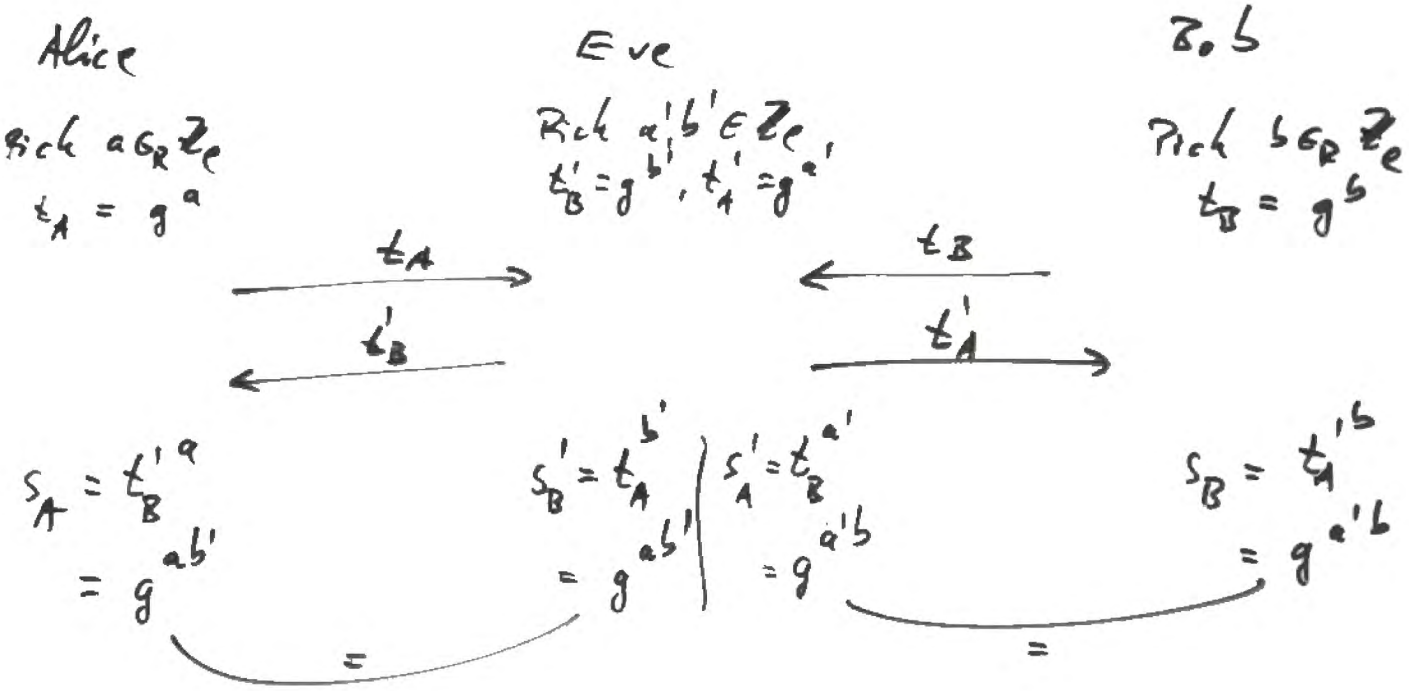
Clearly: if we can solve the DLP then we can solve the DHP.

Converse?

Is that secure?

As long as Eve is passive she cannot find the shared secret w/o solving DHP.

But what if Eve is active?



Usually $s_A \neq s_B$ now ☹

Now any message protected by Alice can be checked and decrypted by Eve and be protected for Bob and so send on.

As a result: Alice and Bob notice nothing and Eve sees everything.

Man-in-the-middle attack

How to prevent that?

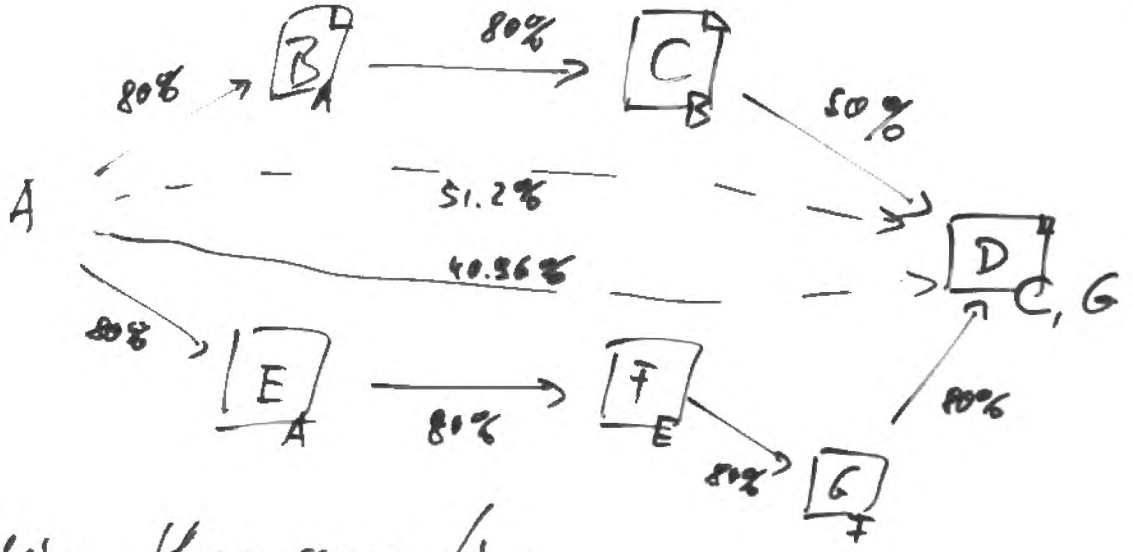
This will be solved later by using public-key signatures on the key exchange after finishing it.

~~How to prevent a man in the middle?~~

(4) How to associate a public key and some identity informations (of a person or entity)?

Solution 1

Web of trust like in PGP, GnuPG.

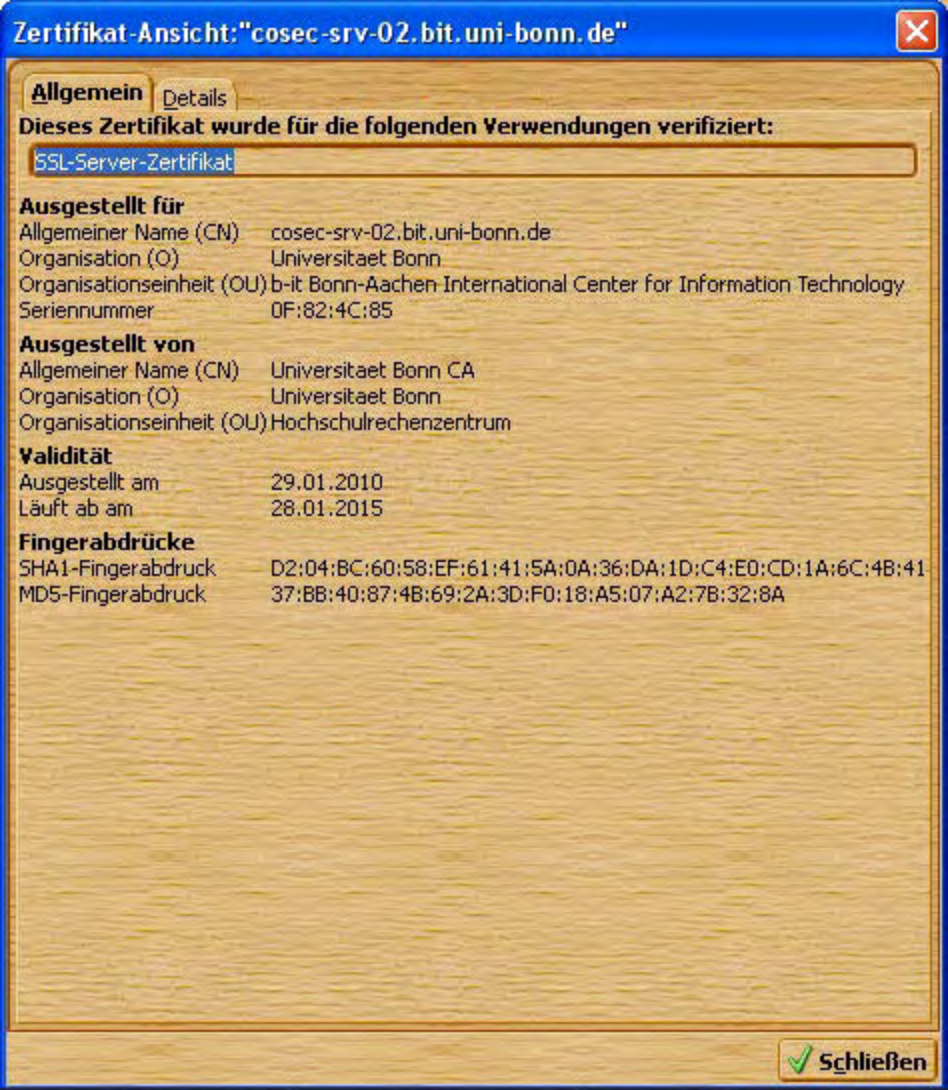


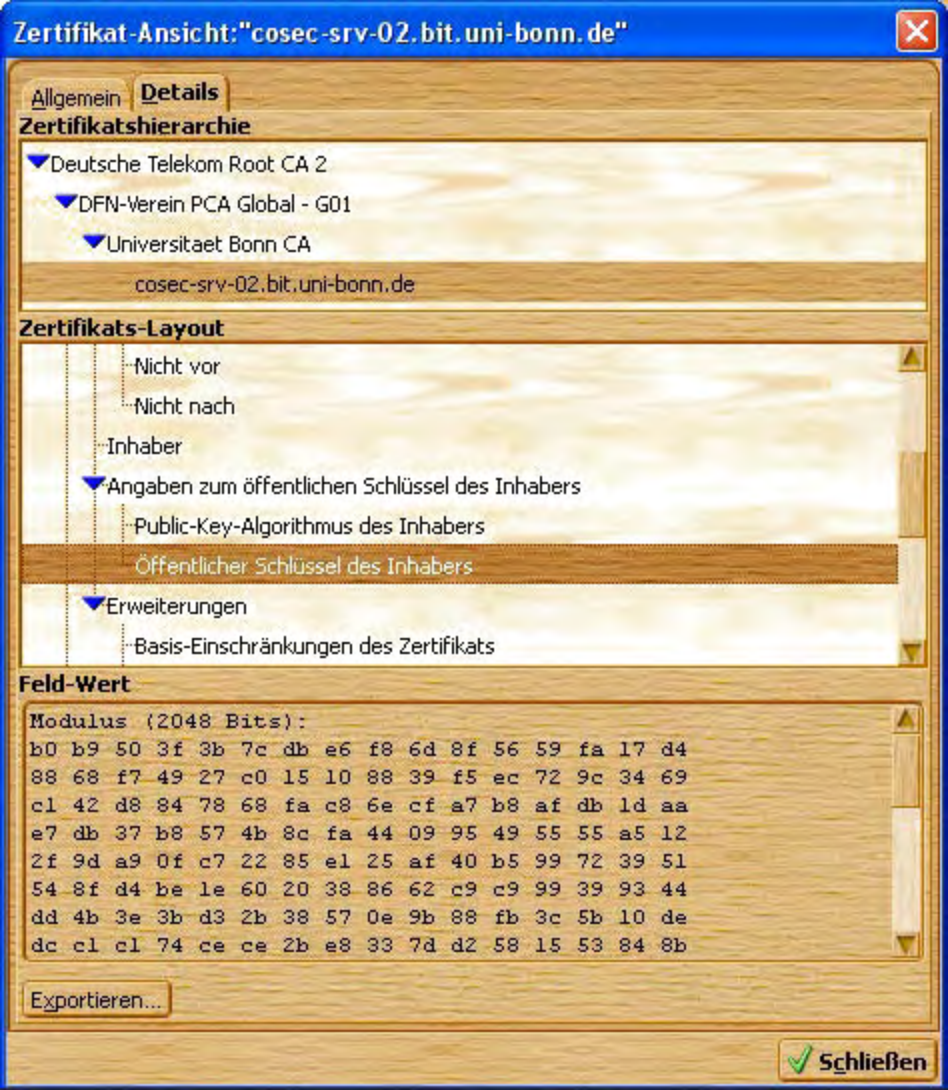
Using these connections
A may give the identity of D some trust...

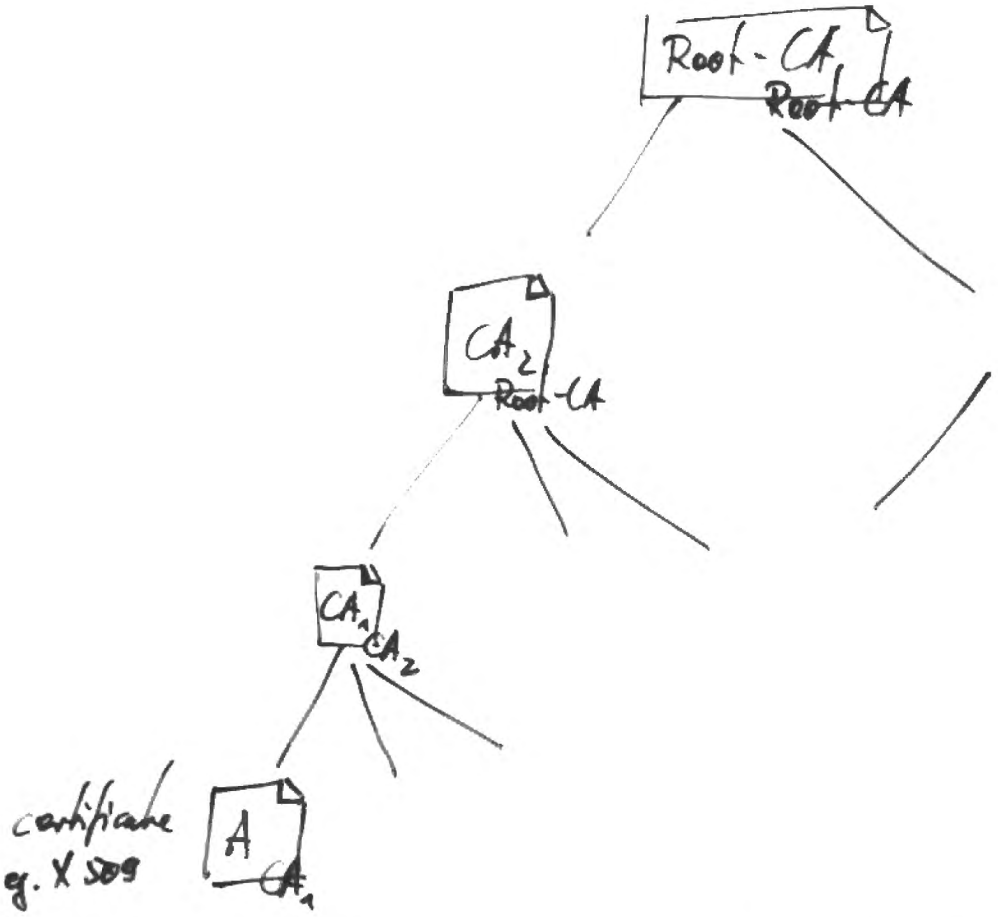
Maybe we would attribute 51.2% + 40.96% of trust to D.

Solution 2

Public Key Infrastructure


Zertifikat-Ansicht:"cosec-srv-02.bit.uni-bonn.de"
Allgemein
Details
Dieses Zertifikat wurde für die folgenden Verwendungen verifiziert:
SSL-Server-Zertifikat
Ausgestellt für
Allgemeiner Name (CN) cosec-srv-02.bit.uni-bonn.de
Organisation (O) Universitaet Bonn
Organisationseinheit (OU) b-it Bonn-Aachen International Center for Information Technology
Seriennummer 0F:82:4C:85
Ausgestellt von
Allgemeiner Name (CN) Universitaet Bonn CA
Organisation (O) Universitaet Bonn
Organisationseinheit (OU) Hochschulrechenzentrum
Validität
Ausgestellt am 29.01.2010
Läuft ab am 28.01.2015
Fingerabdrücke
SHA1-Fingerabdruck D2:04:BC:60:58:EF:61:41:5A:0A:36:DA:1D:C4:E0:CD:1A:6C:4B:41
MD5-Fingerabdruck 37:BB:40:87:4B:69:2A:3D:F0:18:A5:07:A2:7B:32:8A
Schließen


Zertifikat-Ansicht:"cosec-srv-02.bit.uni-bonn.de"
Allgemein
Details
Zertifikatshierarchie
Deutsche Telekom Root CA 2
DFN-Verein PCA Global - G01
Universitaet Bonn CA
cosec-srv-02.bit.uni-bonn.de
Zertifikats-Layout
Nicht vor
Nicht nach
Inhaber
Angaben zum öffentlichen Schlüssel des Inhabers
Public-Key-Algorithmus des Inhabers
Öffentlicher Schlüssel des Inhabers
Erweiterungen
Basis-Einschränkungen des Zertifikats
Feld-Wert
Modulus (2048 Bits):
b0 b9 50 3f 3b 7c db e6 f8 6d 8f 56 59 fa 17 d4
88 68 f7 49 27 c0 15 10 88 39 f5 ec 72 9c 34 69
c1 42 d8 84 78 68 fa c8 6e cf a7 b8 af db 1d aa
e7 db 37 b8 57 4b 8c fa 44 09 95 49 55 55 a5 12
2f 9d a9 0f c7 22 85 e1 25 af 40 b5 99 72 39 51
54 8f d4 be 1e 60 20 38 86 62 c9 c9 99 39 93 44
dd 4b 3e 3b d3 2b 38 57 0e 9b 88 fb 3c 5b 10 de
dc c1 c1 74 ce ce 2b e8 33 7d d2 58 15 53 84 8b
Exportieren...
Schließen



The security of such a PKI lies in:

- the identity of the RootCA and its public key — HUMAN factor
- the care each CA puts in the verification before signing a certificate. — HUMAN factor
- the security of the end-user's computing platform. — HUMAN FACTOR
- the care each CA puts in the protection of its private key, in particular, it must be not be lost or stolen. — HUMAN factor
- the security of the signature scheme(s).

Solution 3 → Identity based cryptography, certificateless schemes.

30.4.13
es
(4)

Security for (2) - (4)

Signatures (2):
Task of the attacker

EUF — Existential unforgeability

The attacker shall produce a valid signed message of his choice for an (uncompromised) user of his choice.

Means of the attacker

CMA — Chosen Message Attack

The attacker gets an oracle that signs any message on behalf of any user.
Possibly, the attacker also gets an oracle that reveals the secret key of any user.

Winning condition:

The attacker must fulfill

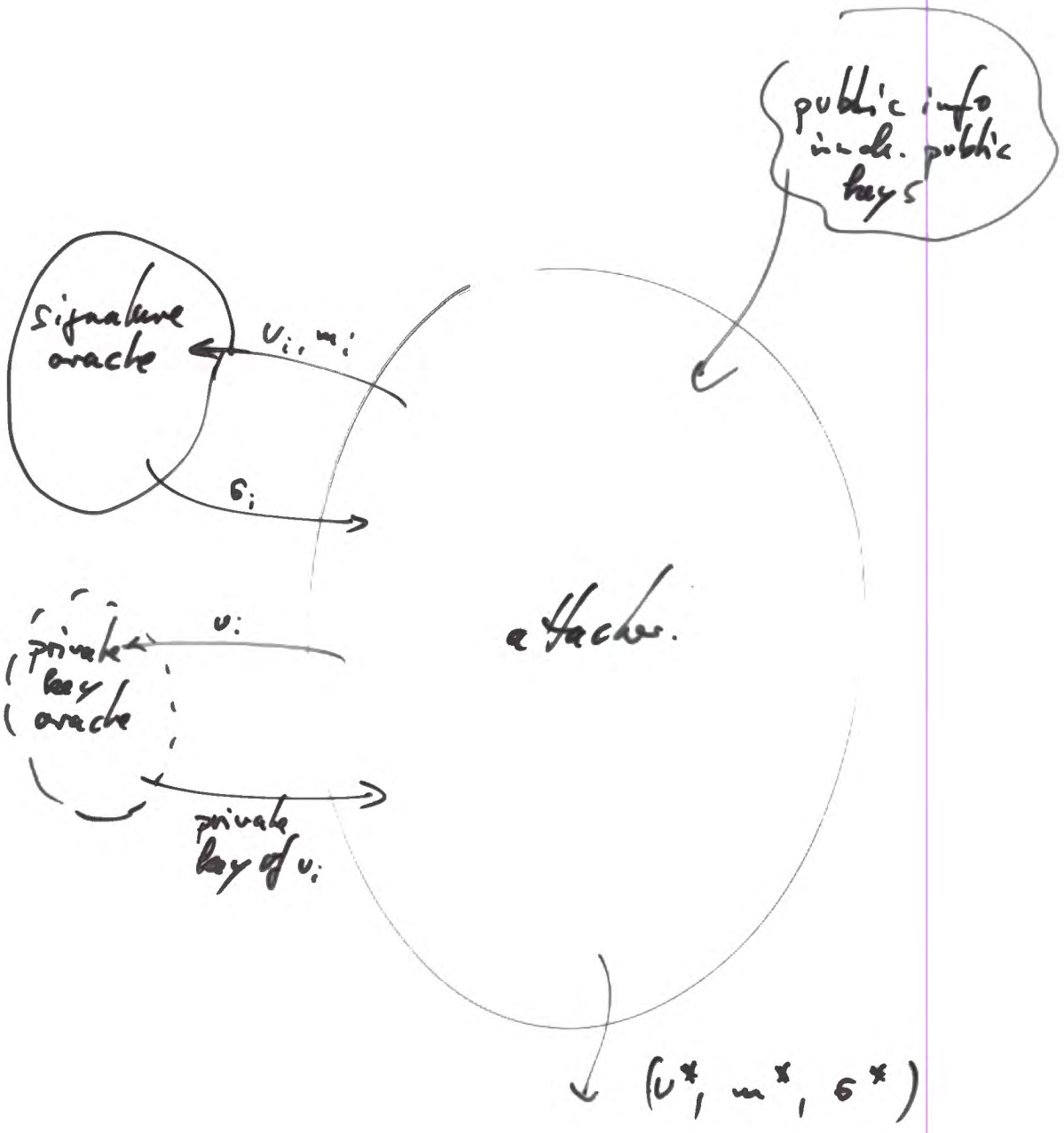
- the task
- but not with trivial data

and all that within poly. time (or fixed time)
and with non-negligible (or fixed/lower-bounded) advantage.

30.4.13
CS
(2)

EUF-CMA secure

There is no such attacker.



RSA-signatures

sign: Input: (N,d), m
Output: σ

1. $hash(m) \in \mathbb{Z}_N$
2. $\sigma \leftarrow hash(m)^d \in \mathbb{Z}_N$
3. return (m, σ)

verify: Input: (N,e), m, σ
Output: valid/invalid.

1. return $\boxed{hash(m) = \sigma^e \text{ in } \mathbb{Z}_N}$.

Theorem
Schoolbook RSA signatures, ie. with hash = id, are not EUF-(CMA) secure. KO

Pf (Dimna)
Pick $\sigma \in_R \mathbb{Z}_N$, compute $m = \sigma^e \in \mathbb{Z}_N$.
This is a **Bleichenbacher** forgery. □

Theorem Schoolbook RSA is not even UUF-CMA secure. △

20.4.13 cS

"**Theorem**"

FDH-RSAsign is .... secure unless

↑ full domain hash.

the hash function is ... insecure or RSA is insecure.

Theorem

If RSA-sign with some function hash is (EUF-CMA.) secure then hash must be

(i) one-way (pre-image-resistant)

(ii) 2nd preimage resistant

(iii) collision resistant.

Proof (i) We prove the converse. Assume we may find preimages.

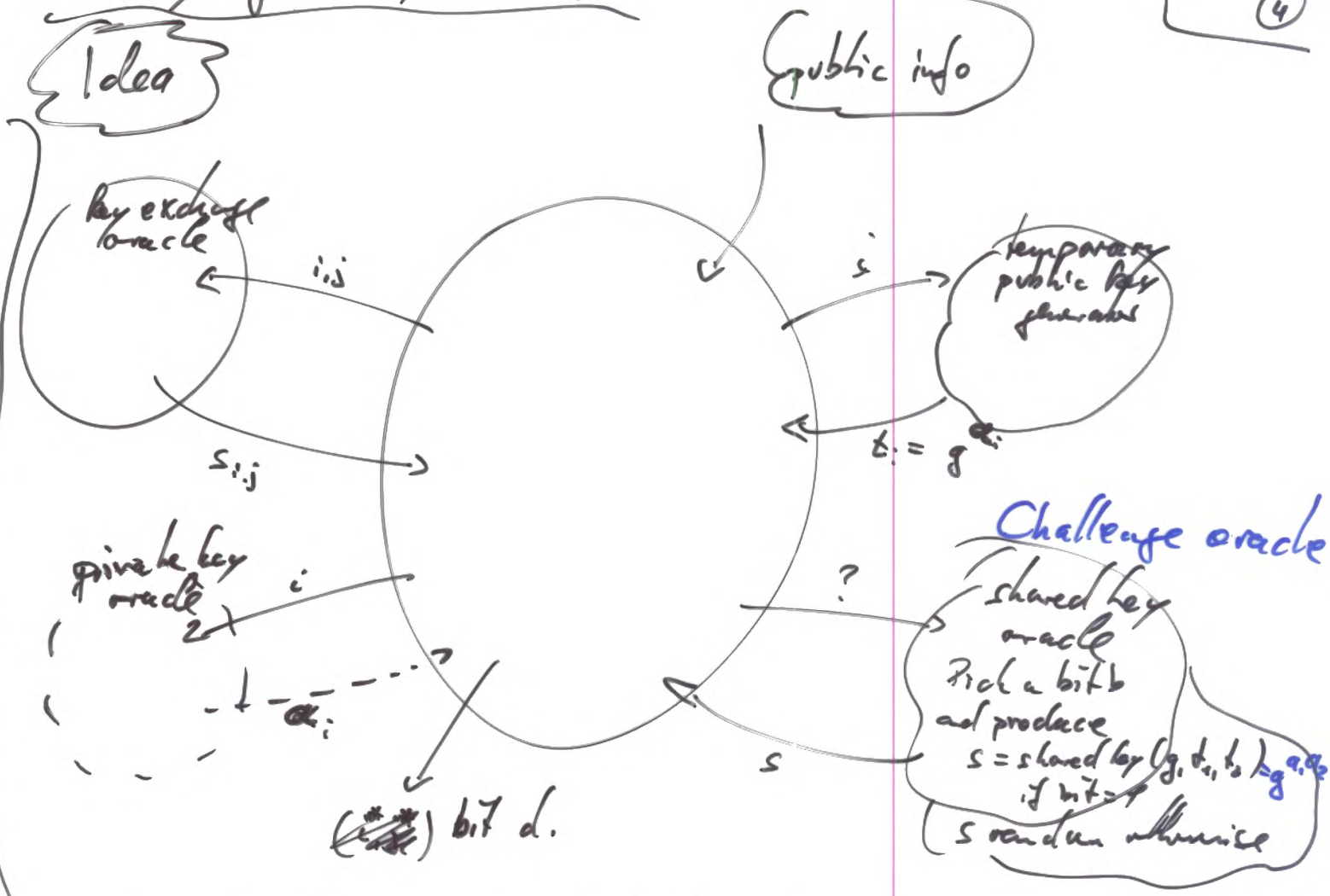
To produce an existential forgery

- pick $\sigma \in \mathbb{Z}_N$.
- compute $\sigma^e \in \mathbb{Z}_N$.
- compute a preimage of σ^e under hash, i.e. $\text{hash}(m) = \sigma^e$
- return (m, σ).

This is an existential forgery (under Key Only Attack). 13

Security of (3) key exchange

Idea



- The attacker's goal shall be to
  - say whether s is the shared key of users 1 and 2, try to make $d = b$.

Theorem?

decisional Diffie-Hellman problem
i.e. given (g, t_1, t_2, s) output 0 or 1 deciding whether $s = g^{a_1 a_2}$ where $t_1 = g^{a_1}$, $t_2 = g^{a_2}$.

If DDH is hard
then there is no such attacker with...

Pf: ... "routine" □

This notion does not exclude man-in-the-middle!

Some examples

- RSA-signature with hash=id.
  is not even UUF-CMA secure.

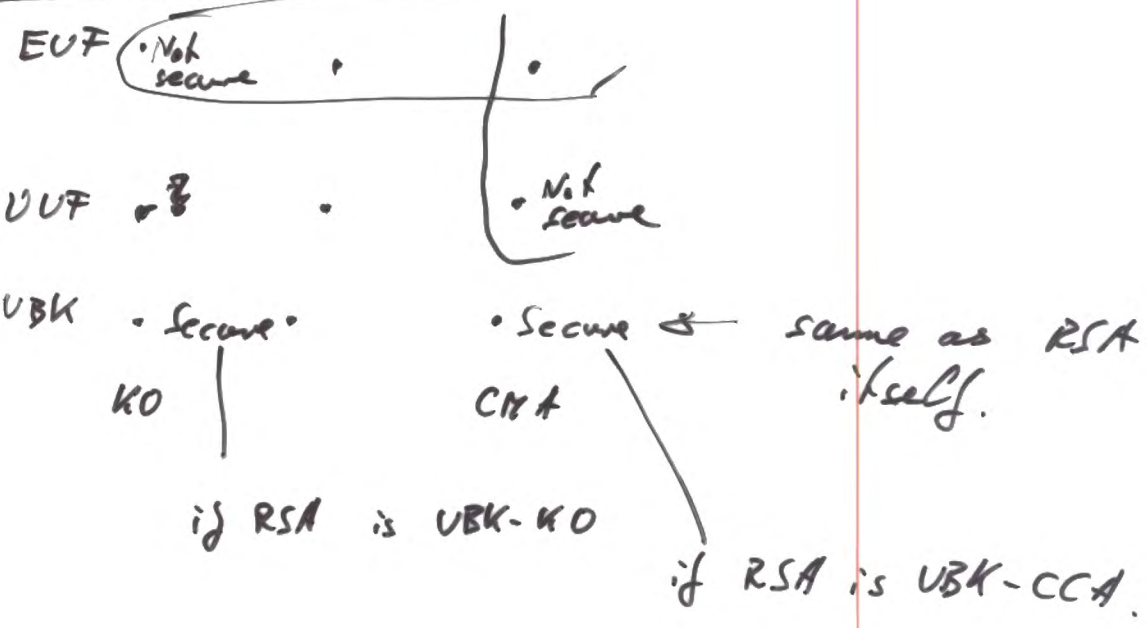
Pf

Attacker — public key (N, e)
Input: message $m \in \mathbb{Z}_N$ — hash(m)
Output: signature σ to m, i.e. "$m^e = \sigma^e$ in $\mathbb{Z}_N$".

1. Pick $m_1, m_2 \in_R \mathbb{Z}_N^*$ such that $m = m_1 m_2$.
2. Ask the CMA oracle for signatures σ_1, σ_2 on m_1, m_2, respectively.
3. Return $\sigma_1 \cdot \sigma_2$.

Even better: pick $m_2 = \sigma_1^e$ and ask CMA oracle only for a signature of $m_2 = m/m_1$. ∎

RSA sign with trivial hash



- RSA-FDH with a hash function that may produce any value in $\mathbb{Z}_N^*$ and is one-way.
  Assuming RSA is UBK-KO secure, then it is EUF-CMA secure if ... (~~see literature~~)

Side remark:

(1) If RSA is broken i.e. not UBK - ~~CMA~~ KO
then we have an ~~EUF~~ - ~~CMA~~ KO attacker.
And any such attacker is also a EUF-CMA attacker.

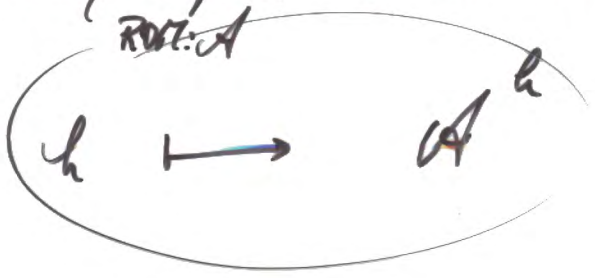
(2) If hash is not one-way
then we have a EUF-KO attacker.
In particular, if is a EUF-CMA attacker.

Actually, we have

Theorem

RSA-FDH is EUF-CMA secure
if RSA is ... secure
~~and h is one and one-way~~
in the Random Oracle Model.

That means the attacker does only get an oracle for computing hash values, rather than the code of the hash function or rather that it may depend on the hash function.



Examples

- RSA-encryption is not IND-CCA secure,
  not even IND-KO secure.

Pf Construct an attacker:

Pick $x_0, x_1 \in_R \mathbb{Z}_N^\times \setminus \{0, \pm 1\}$, $x_0 \neq x_1$.
Ask the challenge oracle for the
encryption y^* of one of them.
Compute $y_i = \text{enc}(x_i)$.

If $y^* = y_0$ answer 0,
if $y^* = y_1$ answer 1,
otherwise tell the challenge oracle a cheater. ☐

This attacker has advantage $\frac{1}{2}$, i.e. best possible. ☐

⇒ More general: no deterministic encryption can be IND-KO secure.

- ElGamal-encryption: see exercise.

  is not IND-CCA secure either.

This is due to the fact that ElGamal encryption is homomorphic:

$$x \longmapsto (g^t, x \cdot a^t) \quad \text{for same random } t.$$

$$1 \longmapsto (g^{t'}, 1 \cdot a^{t'})$$

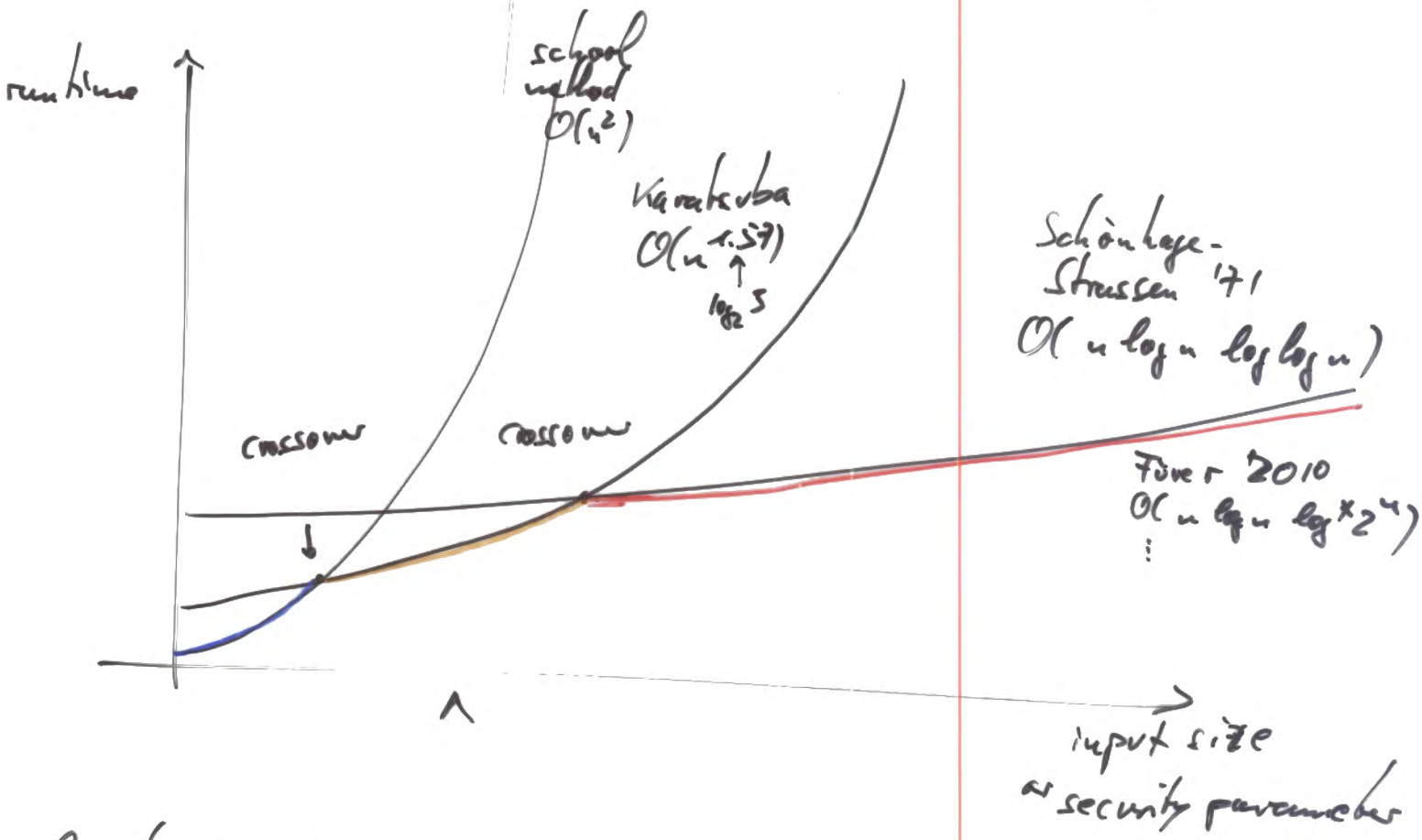
multiply:

$$x \cdot 1 \longmapsto (g^{t+t'}, x \cdot 1 \cdot a^{t+t'})$$

a different ciphertext for the same plaintext.

Remark on efficiency

asymptotic view | example: multiplication of integers



fixed size view

Remark on hybrids

Combining RSA with 2048 bit ≙ 128 bit security
with AES with 128 bit key ≙ 125 bit security

Simple combination: only 64 bit security.
(or 80 bit)

8.5.13
es (2)

Remark on bits

RSA hardcore bit

Assume the attacker has an algorithm B which on
- inputs: ciphertext y, public key (N, e).
- output: $BIT_0(x)$.

This shows that bit decisions may be as powerful as a complete break.

Claim: Using B the attacker can find x!

How to?

Assume $BIT_0(x) = 0$, i.e. $B(y, (N, e)) = 0$.

Then $y_1 = enc(x/2)$

$= enc(x) \cdot \underline{enc(1/2)}$

$2^{-e} \in \mathbb{Z}_N$.

$= y \cdot 2^{-e} \in \mathbb{Z}_N$.

and $x/2$ has just the bits of x shifted to the right by 1 position.

Recursively, we obtain $x/2$ from y_1.

If $BIT_0(x) = 1$, i.e. $B(y, (N, e)) = 1$,

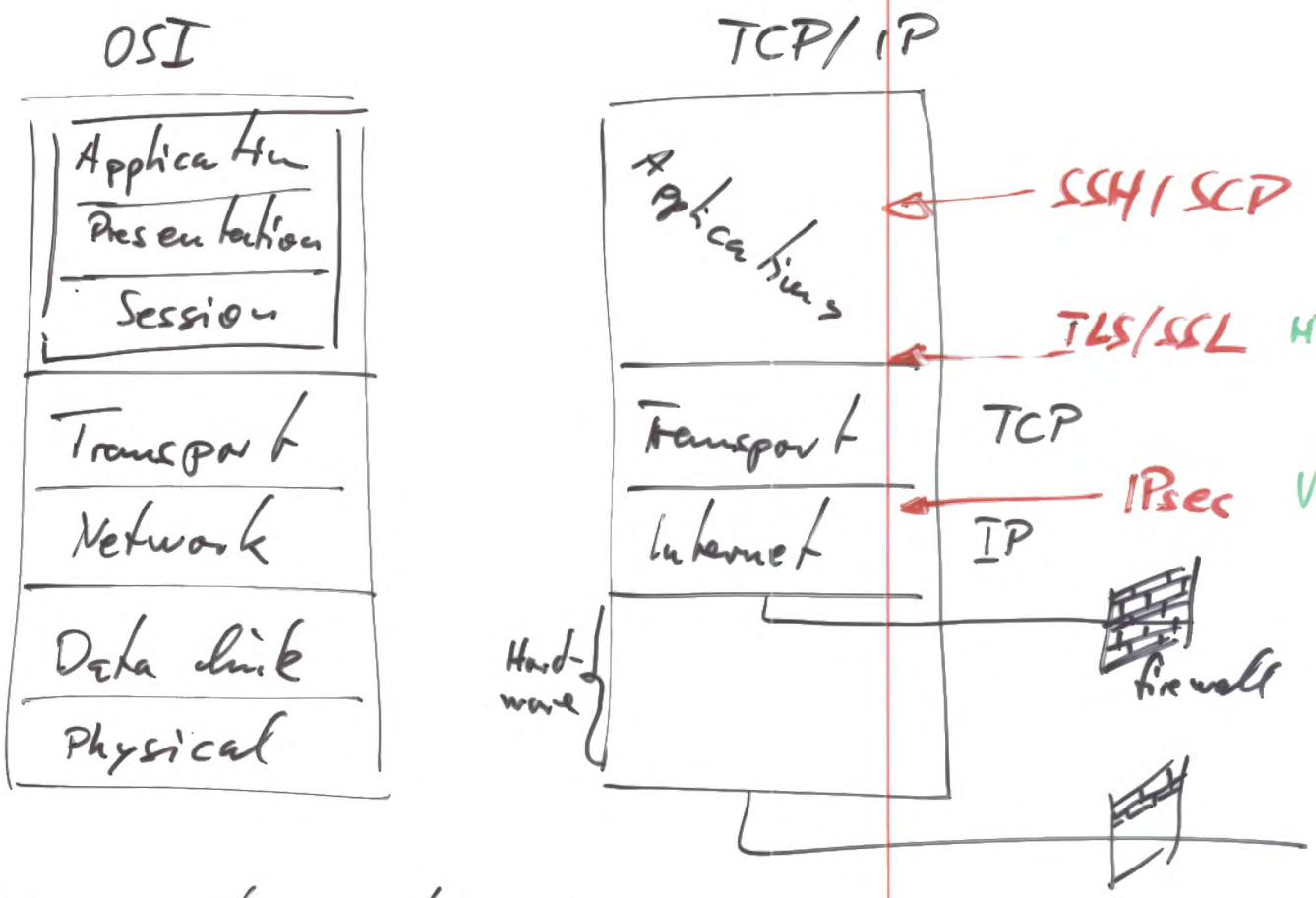
then use $y_1 = enc((N-x)/2) = y \cdot (-2)^{-e} \in \mathbb{Z}_N$.

Now find $(N-x)/2$ from y_1 recursively...

This can even work if B only has some advantage over guessing...

8.5.13
es
(3)

Secure connections in real world deployments



Placement protocols

- The lower — the more traffic is protected.
- The lower — the more software/can use it does
- The lower — the less changes to applications are necessary.
- The ~~higher~~ — the more control inside the applications.
- The lower — the more difficult for the firewall to do its work.

14.5.13
es
(1)

IPsec

is a combination of

IKE = Internet Key Exchange

old: IKEv1

new: IKEv2

comprises:

(1) the exchange of a seed key,

(2) the authentication of the communication partners,

(0) the choice of algorithms and further parameters.

IPsec:

→ AH: authentication header
→ authenticate data and some parts of the IP header.

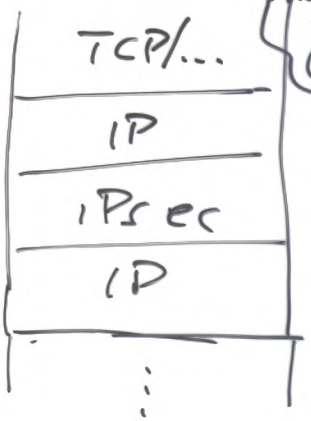
→ ESP: encapsulated security payload
→ allows encryption
→ also allows authentication (w/o header)

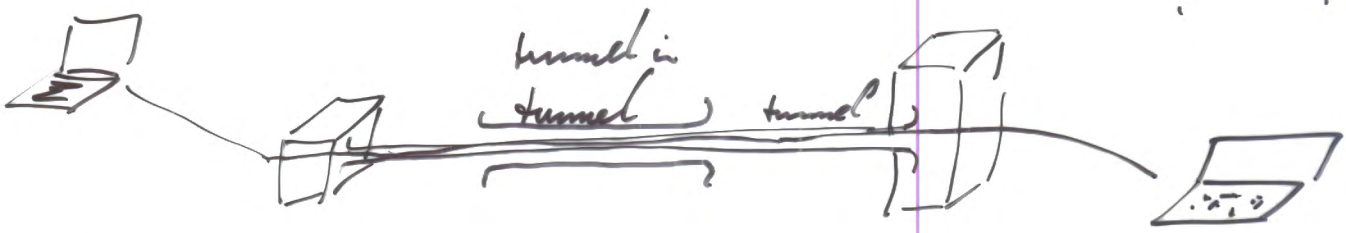
Two ~~transfer~~ modes:

- transport mode:

[IP header | IPsec header | data]

- tunnel mode





Problems

IPsec ESP tunnel-mode

- Packet segmentation and reassembly if original packet too close to threshold size.
- NAT is problematic.
- No stateful inspection of higher layer info.

too many options.

Side remark:

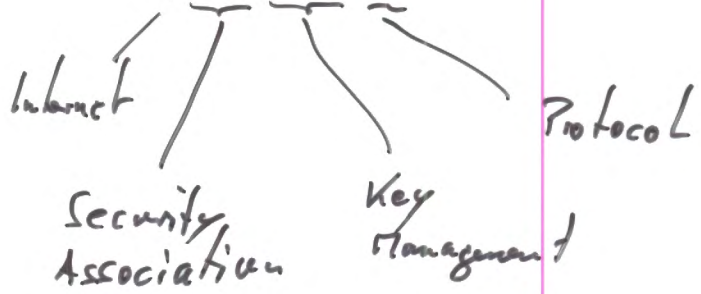
IPv4 is independent of IPsec
but can be used together.

IPv6 always comes with IPsec.

History of IKE

PHOTURIS ⚡ SKIP

NSA proposed ISAKMP



- not really a protocol but only a framework.
- it ruled out both PHOTURIS and SKIP

Pro: • finally a solution was found (IKEv1)

Con: • no clear design
• too many variants
• badly documented, ≥ 3 RFCs, notation is inconsistent (even within single documents)

Development took much too long.

→ IPsec came late

→ IPv6 came late.

IPSEC & IKE

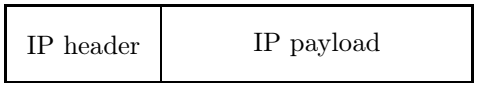
Michael Nüsken

25 June 2007

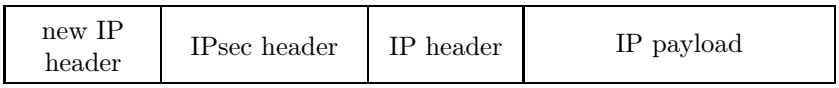
Before all: we are talking about a collection of protocols. Each partner of the exchange has to keep some information on the connection. This is in our context called the security association (SA). It contains specification about the algorithms that should be used for encryption and authentication, it contains keys for these, it may contain traffic selectors (filtering rules), and more. Each SA manages a simplex connection for one type of service. In each direction there will be an SA for the key exchange (IKE_SA) and one for the encapsulating security payload or for the authentication header. So each partner has to maintain at least four SAs. Such an SA is selected by an identifier, the so-called security parameter index (SPI). It is chosen randomly but so that it is unique.

1. IPsec

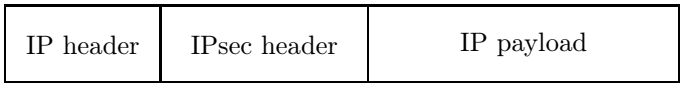
The secure internet protocol modifies the internet protocol slightly. We have the choice between transport and tunnel mode. In tunnel mode, an IP packet



is wrapped in with a new IP header and an IPsec header to

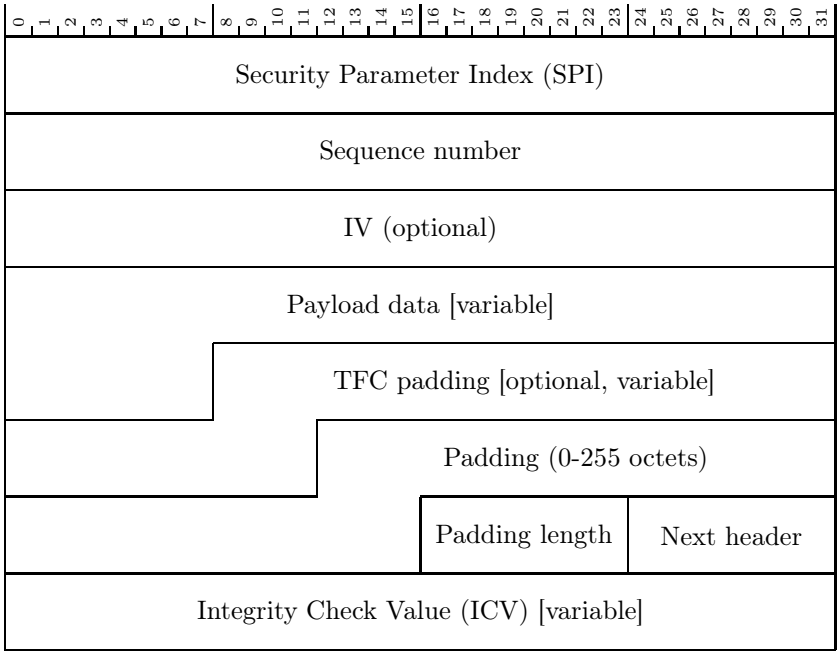


In transport mode, only the IPsec header is added:



There are two types of IPsec headers: the encapsulating security payload (ESP) and the authentication header (AH).

1.1. IPsec encapsulating security payload. The ESP specifies that and how its payload is encrypted and (optionally) authenticated. Actually, this 'header' is split into a part before and one after the data:



The security parameter index identifies the SA and thus all necessary algorithms and key material. To create the secured packet from the original one, it is first padded. Padding is used to enlarge the data length to a multiple of a block size that might be associated with the encryption. Traffic flow confidentiality (TFC) padding can be used to disguise the real size of the packet. Then the data is encrypted; in tunnel mode including the old IP header. To be precise, all the information from Payload data to Next header is encrypted. Next, a message authenticion code is calculated for this encrypted text and security parameter index, sequence number, initialization vector (IV) and possibly further padding; actually the message authentication code covers the entire packet but the header and the integrity check value plus the extended sequence number and integrity check padding if any.

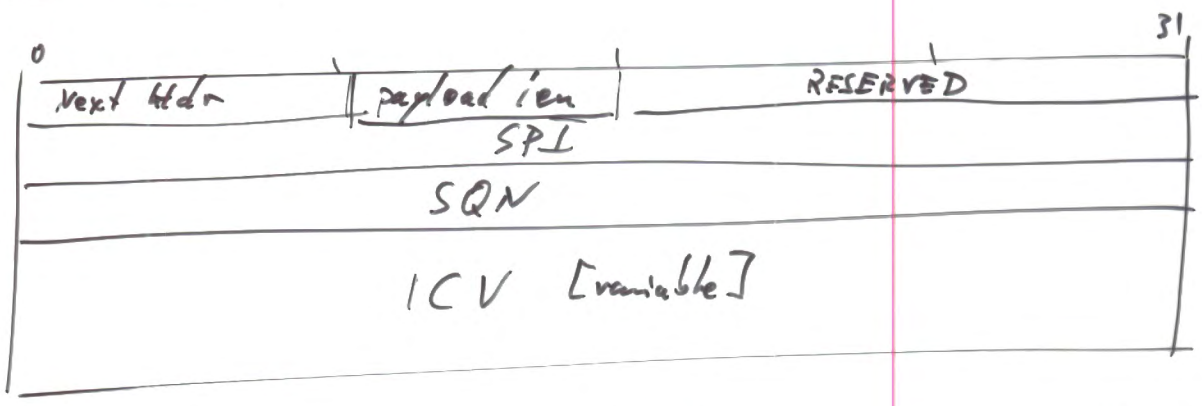
1.2. IPsec authentication header. The AH authenticates its payload and also parts of the IP header. (Yes, this does violate the hierarchy.)

2. Internet key exchange (version 2)

Any message in the internet key exchange starts with a header of the form

IPsec AH

19.5.13 (es) 4

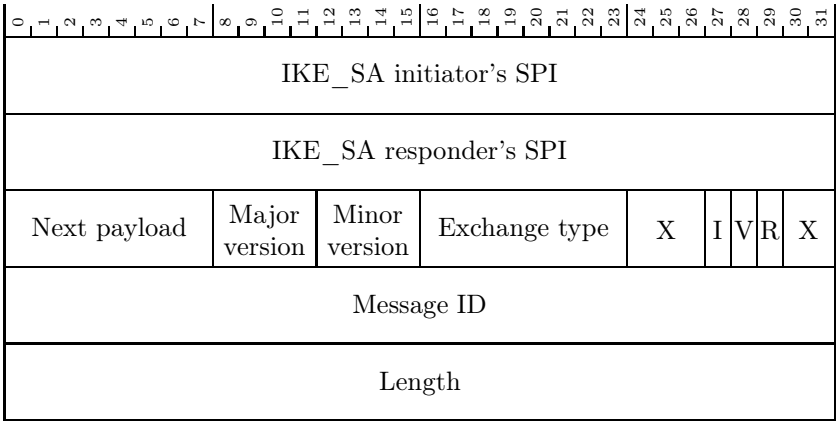


Types of IP header fields: invariable, variable but predictable, or unpredictable.

IP header fields	types
version	invariable
internet header length	
total length	
identification	
protocol	
source address	
destination address	either invariable (w/o routing) or predictable
DSCP	unpredictable
ECN	
Flags	
Fragment offset	
Time To Live (TTL)	
Header Checksum	

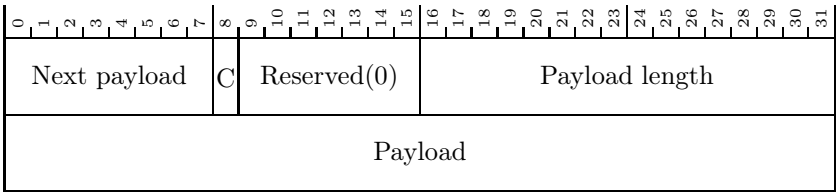
Notes on both ESP and AH

- ESP in earlier version did not have authentication.
- There now is an algorithm to identify the SPI
- Amendment allows to extend sequence numbers to 64bit.
- TFC padding is new.
- New algorithms are added ... in particular, so called AEAD schemes (authenticated encryption with associated data)



Clearly, the version is 2.0 with the present drafts (major version: 2, minor version: 0). The flags X are reserved, the I(nitiator) bit is set whenever the message comes from the initiator of the SA, the V(ersion) bit is set if the transmitter can support a higher major version, the R(esponse) bit is set if this message is a response to a message with this Message ID. The header is usually followed by some payloads like

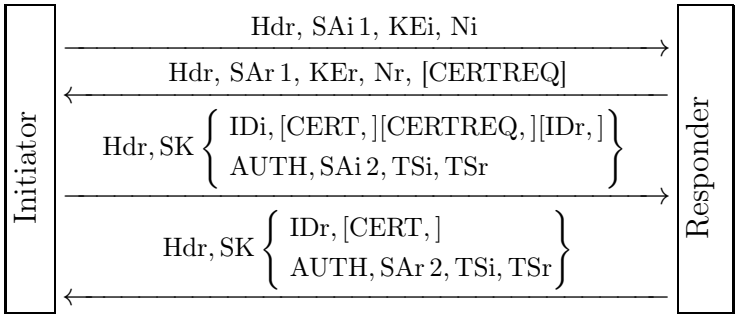
Exchange type	Value
Reserved	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
Reserved to IANA	38-239
Reserved for private use	240-255



The C(ritical) bit indicates that the payload is critical. In case the recipient does not support a critical payload it must reject the entire message. A non-critical payload can be simply skipped. All the payloads defined in RFC4306 are to be handled as critical ones whatever the C bit says.

Next payload	Notation	Value
None		0
RESERVED		1-32
Security Association	SA	33
Key Exchange	KE	34
Identification - Initiator	IDi	35
Identification - Responder	IDr	36
Certificate	CERT	37
Certificate Request	CERTREQ	38
Authentication	AUTH	39
Nonce	Ni, Nr	40
Notify	N	41
Delete	D	42
Vendor ID	V	43
Traffic Selector - Initiator	TSi	44
Traffic Selector - Responder	TSr	45
Encrypted	E	46
Configuration	CP	47
Extensible Authentication	EAP	48
Reserved to IANA		49-127
Private use		128-255

2.1. Initial exchange.



Protocol 2.1. IKE_SA_INIT.

1. Prepare SAi1, the four lists of supported cryptographic algorithms for Diffie-Hellman key exchange (groups), for the pseudo random function used to derive keys, for encryption, and for authentication.
   Guess the group for Diffie-Hellman and compute $\mathrm{KEi} = g^a$.
   Choose a nonce Ni.
   $\xrightarrow{\text{Hdr, SAi 1, KEi, Ni}}$
2. Choose SAr1 from SAi1 unless no variant is supported.

Compute KEr $= g^b$ if the group was guessed correctly. (Otherwise send:

$$\text{Hdr}, \text{N}(\texttt{INVALID\_KE\_PAYLOAD}, \text{group})$$

.)
Choose a nonce Nr.

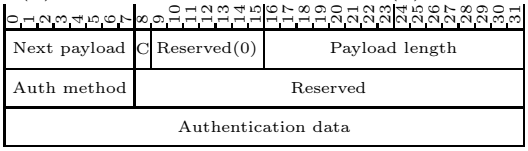
$$\xleftarrow{\text{Hdr}, \text{SAr 1}, \text{KEr}, \text{Nr}, [\text{CERTREQ}]}$$

3. Both parties now derive the session keys. We assume that prf is the selected pseudo random function which gets a key and a bit string as input.

$$\begin{aligned}&\text{SKEYSEED} = \text{prf}(Ni|Nr, g^{ab}),\\ &\text{SK\_d|SK\_ai|SK\_ar|SK\_ei|SK\_er|SK\_pi|SK\_pr}\\ &= \text{prf+}(\text{SKEYSEED}, \text{Ni} \mid \text{Nr} \mid \text{SPIi} \mid \text{SPIr})\end{aligned}$$

where prf+(K, S) $= T_1|T_2|T_3|\ldots$, and T_1 $=$ prf$(K, S|0x01)$, T_i $=$ prf$(K, T_{i-1}|S|i)$ for $i > 1$. SK_d is used for the derivation of keys in a child SA. SK_ai and SK_ei are used for authenticating and encrypting messages sent by the initiator, SK_ar and SK_er for messages sent by the responder.

4. The initiator send its identity IDi, optionally one or more certificates CERT, a certificate request CERTREQ (possibly including a list of trusted CAs), and optionally the responders identity IDr (it may be that the responder serves multiple identities 'behind' it).
Further she computes an authentication AUTH (using the key from the first CERT payload) for the entire first message concatenated with the responder's nonce Nr and the value prf(SK_pi, IDi). The authentication method can be RSA digital signature (1), shard key message integrity code (2), or DSS digital signature (3).



The initiator starts to negotiate a child SA in SAi 2 with proposed traffic selectors TSi, TSr.

$$\xrightarrow{\text{Hdr}, \text{SK} \left\{ \begin{array}{l} \text{IDi}, [\text{CERT},] \\ [\text{CERTREQ},] \\ [\text{IDr},] \\ \text{AUTH}, \text{SAi 2}, \\ \text{TSi}, \text{TSr} \end{array} \right\}}$$

5. The responder sends its identity IDr, certificate(s). He computes an authentication AUTH for the entire second message concatenated

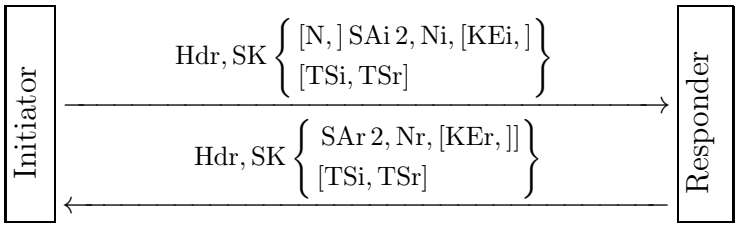
with the initiator's nonce Ni and the value prf(SK_pr, IDr).
Further he supplies the answer SAr 2 to the child SA creation and sends the accepted traffic selectors TSi, TSr.

$$\text{Hdr, SK} \left\{ \begin{array}{l} \text{IDr, [CERT,]} \\ \text{AUTH, SAr 2,} \\ \text{TSi, TSr} \end{array} \right\} \longleftarrow$$

If this initial exchange is completed successfully the IKE_SA and a CHILD_SA are ready for use. Keying material for the childs is generated similar to the IKE_SA keys:

$$\text{KEYMAT} = \text{prf+}(\text{SK\_d}, \text{Ni} \mid \text{Nr})$$

2.2. Creating additional child SAs. Further childs can be created under this IKE_SA using a CREATE_CHILD_SA exhange:



In case a CHILD_SA shall be rekeyed the notification payload N of type REKEY_SA specifies which SA is rekeyed. This can be used to established additional SAs as well as to rekey ages ones. Create new ones and afterwards delete the old ones. Also the IKE_SA can be rekeyed similarly.

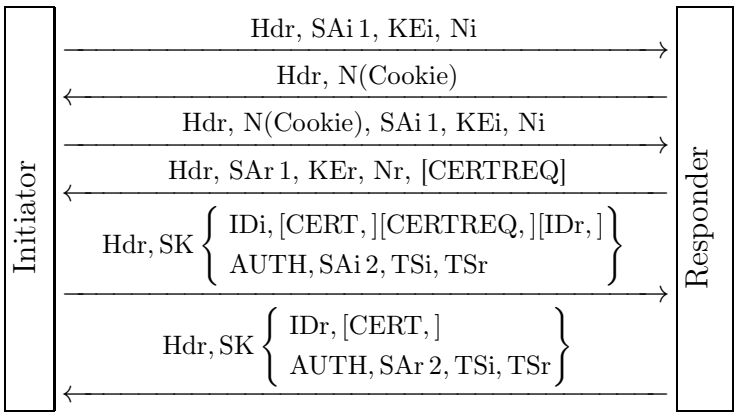
In a CREATE_CHILD_SA exchange including an optional Diffie-Hellman exchange new keying material uses also the new Diffie-Hellman key g^{ir}, it is concatenated left to the nonces. (Though the Diffie-Hellman key exchange is optional, it is recommended to either used it or at least to limit the number of uses of the original key.)

2.3. Denial of Service. If the server has a lot of half open connections (ie. the first message arrived, the second was sent but the third message is pending) it may choose to send a cookie first. (In order to defeat a denial of service attack.) It is suggested to use a stateless cookie consisting of a version identifier and a hash value of the initiator's nonce Ni, her IP IPi, her security parameter index SPIi and some secret:

$$\text{Cookie} = \text{verID} \mid \text{hash}(\text{Ni}, \text{IPi}, \text{SPIi}, \text{secret}_{\text{verID}})$$

This way the secret can be exchanged periodically, say every second, and the server only needs to store the last few (randomly) generated secrets.

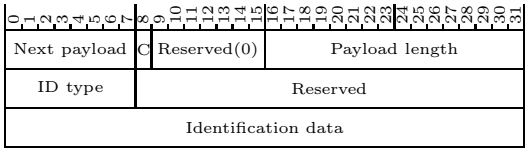
The authentication AUTH then refers to the second version of the corresponding message, so the one including the cookie or responding to that, respectively. So the protocol becomes:



2.4. Extended authentication protocols. The initiator may leave out AUTH and thereby tell the responder that she wants to perform an extensible authentication which is then carried out immediately.

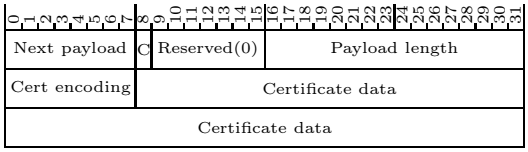
2.5. IP compression. The parties can negotiate IP compression.

2.6. ID payload. The ID payload



can be an IP address (ID type 1), a fully-qualified domain name string (2), a fully-qualified RFC822 email address string (3), an IPv6 address (5), an ASN.1 X.500 Distinguished Name [X.501] (9), an ASN.1 X.500 general name [X.509] (10), a vendor specific information (11).

2.7. CERT payload. The CERT payload



can be encoded in various widely used formats. Note that it can also carry revocation lists.

16.5.13
es
(1)

Securing questions

(0) Securing?

(1) Session key agreement:

- How long? Random? Unpredictable?
- Do both parties contribute to it?
- Men in the middle?

(2) Perfect forward securing „Beagle Boys".

- Can an attacker decrypt recorded conversation given the long-term secrets after termination of the session?

Escrow failage

- Can ---.

secrets before start of the session?

(3) Denial of service.
Does it have serious effects?

(4) Endpoint identifier binding

- Which identifier can a passive or active attacker (possibly impersonating one of the parties) obtain?

(5) Live partner reassurance

- Is replay possible?

(6) Plausible deniability
- Would the log of the session prove?

(7) Stream protection
- → Confidentiality?
- → Integrity?
- → Authenticity?

(8) Negotiating parameters (incl. algorithms)

Add security questions w.r.t. IPsec

28.5.13
es
(1)

(1) Session key agreement

- How long? → look at the group of the DH key exchange.

  Ex:

Group number	Bit length / Type
2	1024 mod p group ← ± 80 bit security
14	2048 mod p group ← ± 100 bit security

  more groups found at IANA.

  The length is big enough to withstand guessing or similar ...

- Random? Unpredictable?

  well, the specs allow KE to be non-random. Then the nonces take over to ensure the randomness.
  The shared key depends on the DH key "g^{xy}" and the nonces N_i, N_r.

  An outside attacker thus cannot tell anything about the next SKEYSEED even he knows all previous ones ~~and maybe also g~~ ...

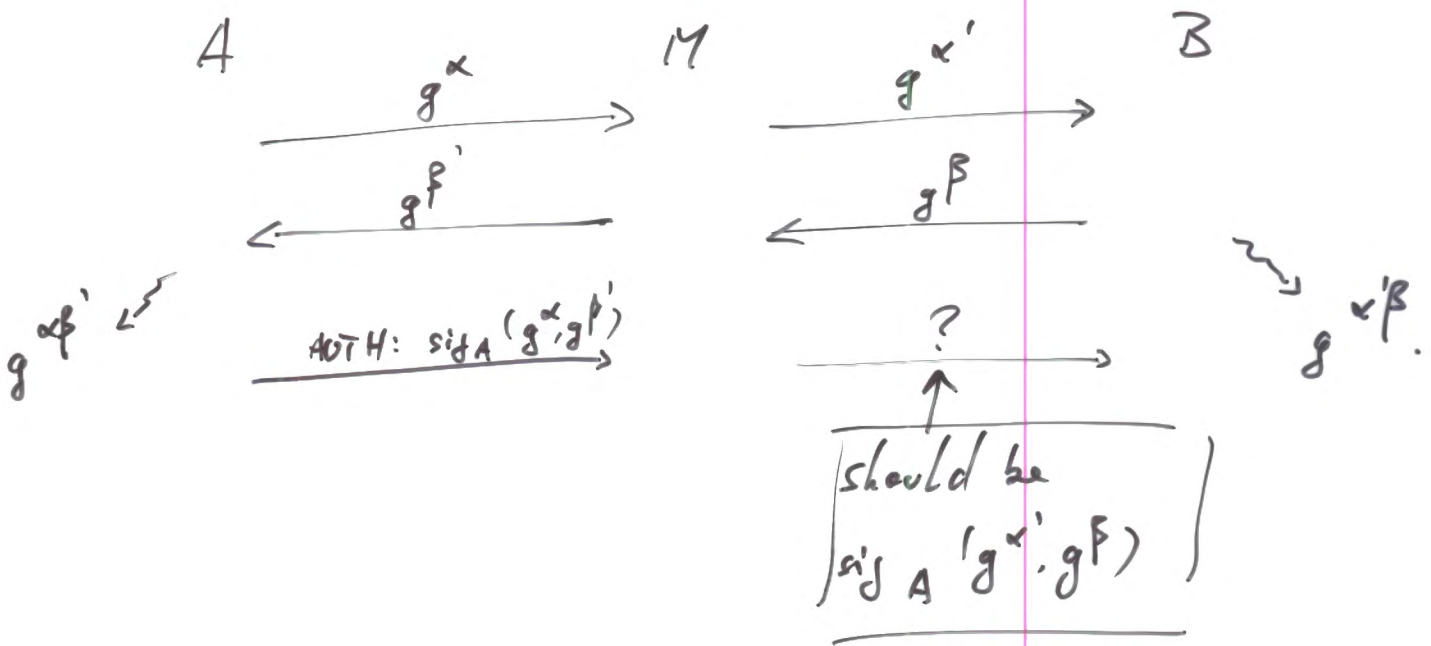
- Do all parties contribute to the key?

  Even if the DH key exchange uses the same a, b then still the nonces ensure new unpredictable keys.

28.5.13
es
(2)

• Alice in the middle ?

Let's try:



So either M forwards $sig_A(g^{\alpha}, g^{\beta'})$

but this only correct if M has forwarded

g^{α} as $g^{\alpha'}$ and g^{β} as $g^{\beta'}$, or at least

$hash(g^{\alpha}, g^{\beta'}) = hash(g^{\alpha'}, g^{\beta})$.

If, say he picks α' ~~at random~~ somehow,

then he still has to solve a second preimage problem for the hash function to ~~get~~ be able to forward the signature, or to break sig_A ...

If he picks $g^{\alpha'} = g^{\alpha}$ he will never know $g^{\alpha\beta}$ unless he breaks a DHP.

There may be intermediate choices...

But if not and (DHP), hash, sig are all hard then M loses.

28.5.13
es
(3)

(1) Perfect forward security "Beagle boys attack"

- Can an attacker decrypt recorded conversations given long-term secrets after termination of the session?

Actually, in IPsec the ~~prime~~ long-term private keys are only used for authentication (in AUTH) but not at all for the shared key SKEYSEED. So you actually do not get extra information.

Escrow failure

- ------ before start of the session?

Assuming that Escrow has the private keys of ~~a~~ one party he cannot do anything.
Of course, if an attacker hacks both parties private keys he may produce both needed signatures and thus play the man in the middle.

Necessary: escrow knows at most one private key or stays passive.

28.5.13
es
(4)

(3) Denial of service

In IPsec we may use stateless cookies.
They ensure that a first IKE_SA_INIT message does induce almost no work and storage for the responder.
The only way the attacker may cause storage use is by revealing the source IP address(es) he uses. This in turn enables network admins to purge or block those connections.

(4) End point identity hiding.

- Which IDs can a passive or active attacker obtain?

Passive / IPsec: the attacker would have to key exchange or encryption to get any info because it is only contained in the protected message of IKE_AUTH..

Active responder / IPsec: Yes, because the initiator sends his ID info first. The attacker does need to reveal C any information on himself.

Active initiator / IPsec: No, unless the attacker reveals his CERT for the AUTH.

Note: it's best possible to hide one of the IDs versus

28.5.15
es
(5)

(5) Live partner reassurance

- Is a replay possible?

IPsec: No, because the keys, SKEYSEED, completely changes if only one of N_i, N_r, g^{ab} is modified.

In a replay the attacker may possibly fix two of them but not all three values.

With new key material none of the old ciphertext messages makes sense in the new session.

4.6.15
es
(4)

(6) Plausible deniability

Since the key material for the MAC (or IC) is known to both parties, we cannot expect to distinguish the two parties as ~~originators~~ writers of certain message but only distinguish the two from third parties.

(7) Stream protection

We (probably) have confidentiality if the used encryption provides it.

We (probably) have integrity and authenticity if the used message authentication code provides it.

But be careful! THAT may fail completely:

Say, enc is a XYZ-secure encryption scheme
↑ e.g. IND-CCA within time t and advantage ε.

and assume $enc_k(p) = p \oplus$ keystream$_k$

Say, mac is a XYZ-secure message authentication scheme.

Construct:

$\widetilde{enc}$: $p \triangleleft \tilde{p} \xrightarrow{enc} \tilde{c}$.

$0 \longmapsto 00$

$1 \longmapsto \{^{01}_{10}$ randomly

We use $mac(p)$ and append it:

$\widetilde{enc}(p)$, $mac(p)$.

Here, confidentiality is completely destroyed...
because of authenticity...

... Attacker flips some bits and learns the entire plaintext (bit by bit) from reactions of the server: OK or ERROR.

(8) Negotiating parameters / algorithms.

4.6.13
es
(3)

The alternative is to fix a particular choice of algorithms.

If your choice is broken tomorrow you have to completely redesign your scheme.

With negotiation broken stuff will just not be chosen or proposed any more.

IPsec: does it.

SSH protocol for Secure Remote login over insecure nk

Runs on top of TCP/IP - Provides: Encryption, Server Auth, integrity prob

DH for KE. After KE, client requests services

Connection: Client... initiates
Server... listens for connection on port 22

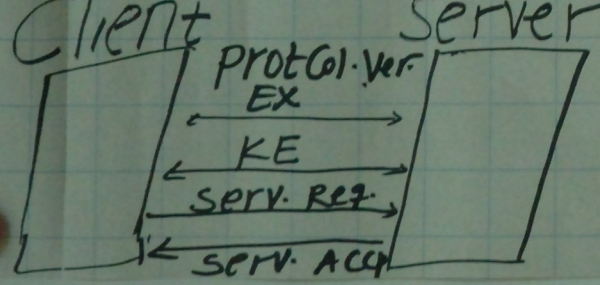
KEY ASSOCIATION — PKI / LOCAL DB

- → GOOD PSEUDORANDOM NUMBER GENERATOR
- → ONLY "STRONG" CIPHERS
- → REKEY TO PREVENT REPLAY ATTACKS
- → ONLY ALLOW SUBSET OF CLIENTS TO CONNECT — DOS
- → DIFFEE HELLMANN FOR PERFECT FORWARD SECRECY
- → DISABLE DEBUG MESSAGES SO AS NOT TO REVEAL CRITICAL INFO

SSH AUTHENTICATION PROTOCOL

- PERFORMS USER CLIENT AUTENTICATION
- RUNS OVER THE SSH TRANSPORT LAYER PROTOCOL
- ASSUMES THAT THE UNDERLYING PROTOCOL PROVIDE: INTEGRITY AND CONFIDENTIALY PROTECTION

AUTHENTICATION METHODS: PUBLIC KEY AUTHENTICATION
PASSWORD AUTHENTICATION
HOST-BASED AUTENTICATION



SSH protocol for Secure Remote login over insecure, nw
Runs on top of TCP/IP - Provides: Encryption, Server Auth, integrity prot

DH for KE. After KE, client requests services

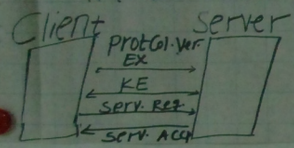
Connection: client ... initiates
server ... listens for connection on port 22

KEY ASSOCIATION — PKI / LOCAL DB

- → GOOD PSEUDORANDOM NUMBER GENERATOR
- → ONLY "STRONG" CIPHERS
- → REKEY TO PREVENT REPLAY ATTACKS
- → ONLY ALLOW SUBSET OF CLIENTS TO CONNECT — DOS
- → DIFFEE HELLMANN FOR PERFECT FORWARD SECRECY
- → DISABLE DEBUG MESSAGES SO AS NOT TO REVEAL CRITICAL INFO

SSH AUTHENTICATION PROTOCOL
- PERFORMS USER CLIENT AUTENTICATION
- RUNS OVER THE SSH TRANSPORT LAYER PROTOCOL
- ASSUMES THAT THE UNDERLYING PROTOCOL PROVIDE INTEGRITY AND CONFIDENTIALY PROTECTION

AUTHENTICATION METHODS:
- PUBLIC KEY AUTHENTICATION
- PASSWORD AUTHENTICATION
- HOST-BASED AUTENTICATION



SSH

connect
authentica
transport

TCP

client
server

user
pwd
pf

server auth
KE
Encryp
integ

payload → Compress

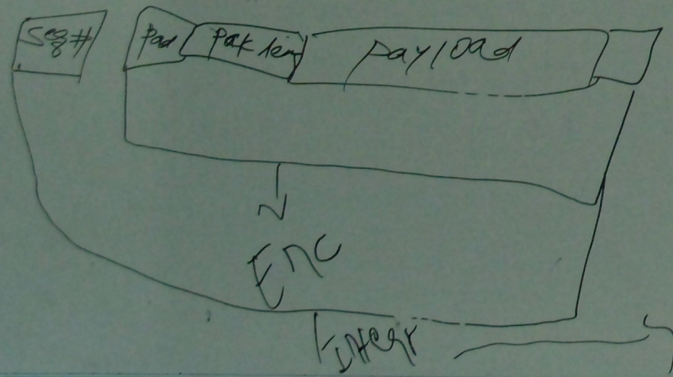
hash(V_C||V_S||I_C||K_S||e||f||K)

DH

KE alg negotiation

Service req

Service accept



SSH-packet

payload
Compre
com pay

x C
e = g^x
y
f = g^y

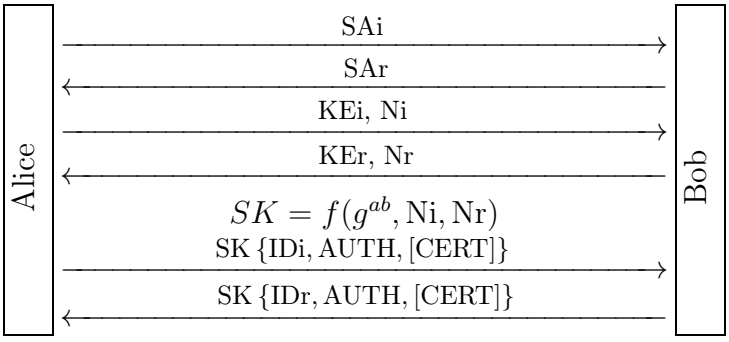
3. IKE version 1

The version 1 of the internet key exchange distinguishes between a main mode and an aggressive mode. Further it allows four variants in each mode depending on the desired type of authentication. Authentication can be based on

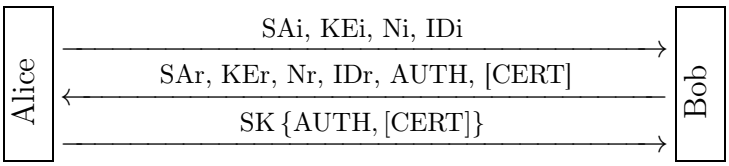
- public signature keys,
- public encryption keys, originial protocol,
- public encryption keys, revised protocol, or
- a pre-shared secret.

We only give the bare protocol summaries here, using notation similar to the one used for version 1. (They are not based on RFC240x but on the book **?**.)

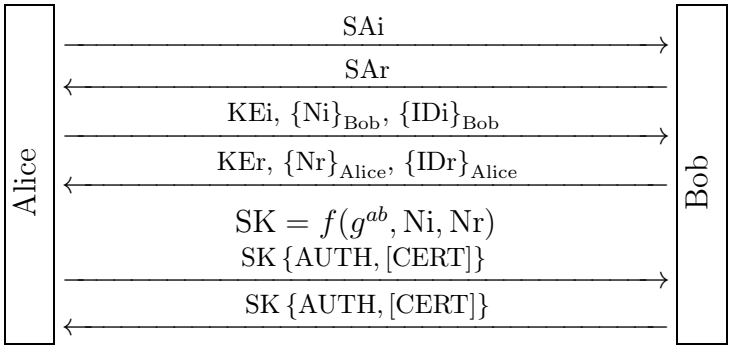
3.1. Main mode, public signature keys.



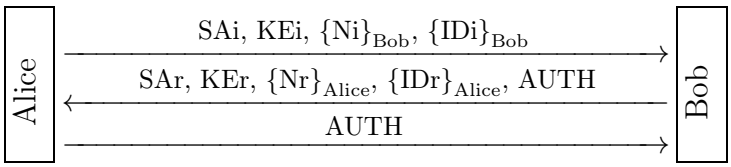
3.2. Aggressive mode, public signature keys.



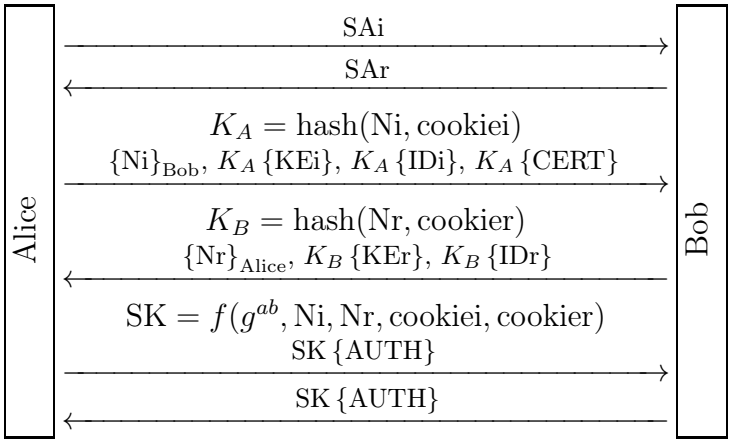
3.3. Main mode, public encryption keys, original protocol.



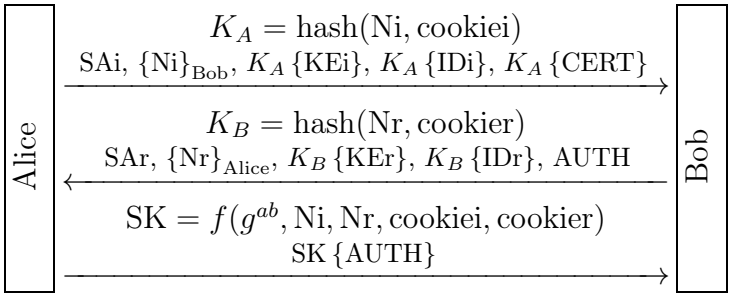
3.4. Aggressive mode, public encryption keys, original protocol.



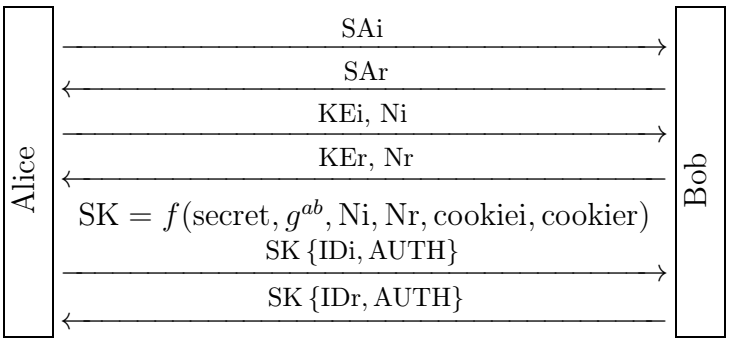
3.5. Main mode, public encryption keys, revised protocol.



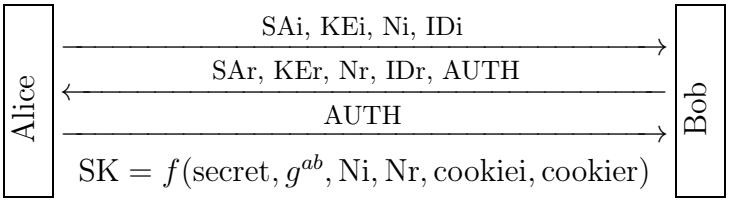
3.6. Aggressive mode, public encryption keys, original protocol.



3.7. Main mode, pre-shared secret.

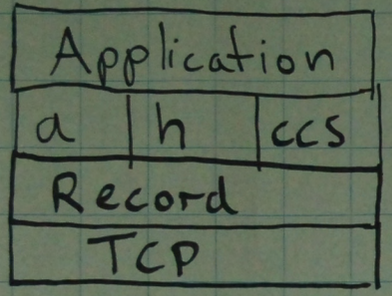


3.8. Aggressive mode, pre-shared secret.

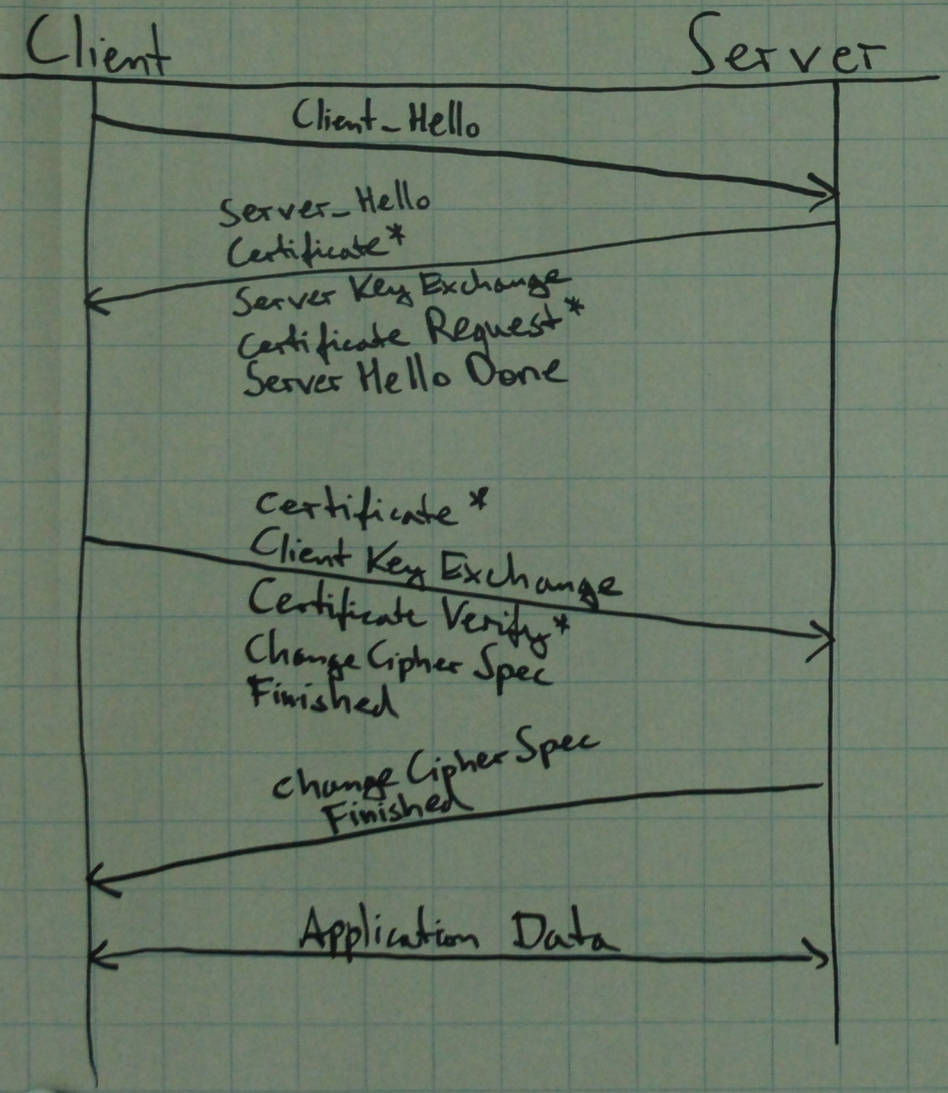


Michael Nüsken
b-it, Bonn, Germany

Transport Layer Protocol (TLS)



Handshake Protocol



Key Exchange

- RSA
- Fixed DH
- Ephemeral DH
- Anonymous DH

⇓

pre shared key

⇓

master_secret
= PRF(pre Shared key,
ClientHello.random+
ServerHello.random)
[0..47]

Security Analysis

- Man in the middle
  ↳ Possible for Anonymous DH
- Mutual authentication
  ↳ Possible if both parties provide cert.
- Perfect Forward Security
  ↳ Only for DH
- Denial of Service
  ↳ not provided (RSA)
- Endpoint Identifier Hiding
  ↳ not provided
- Live Partner Reassurance
  ↳ Yes, due to nonces

Have now seen:

IPsec

SSH

TLS/SSL

Some addons:

Algorithm for IPsec

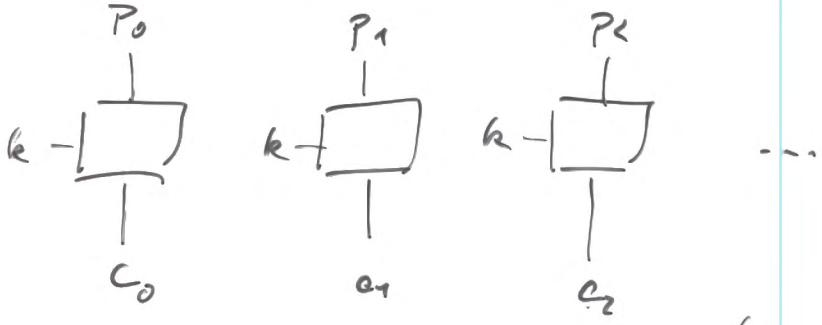
http://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xml

For encryption of long messages using a block cipher

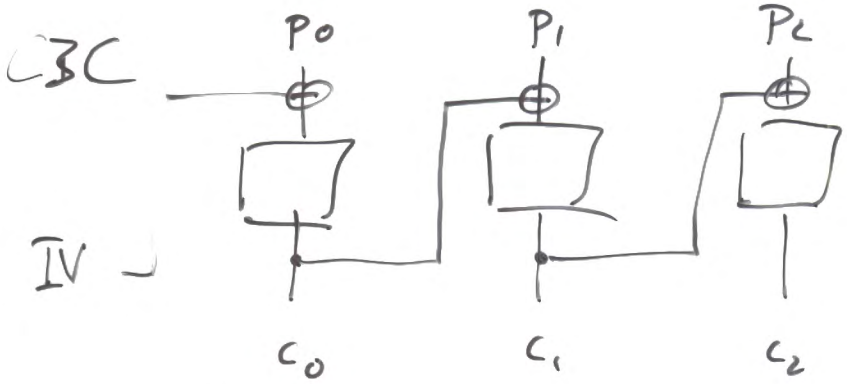
12.6.13
es
(2)

~~CBC mode:~~

ECB mode (electronic code book):

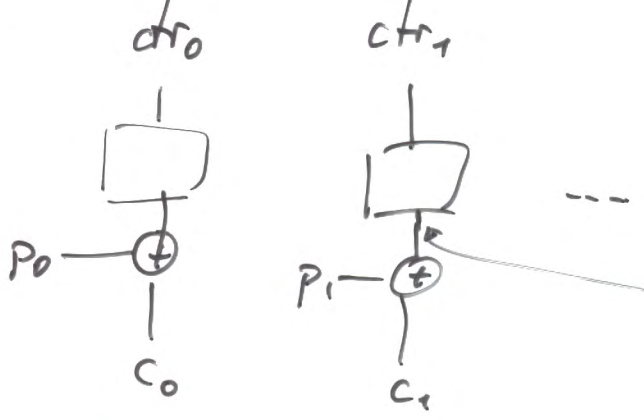


Don't use it. → Wikipedia for an example
→ easy to exchange blocks



good if; provably if IV is fixed & block cipher is good.

CTR



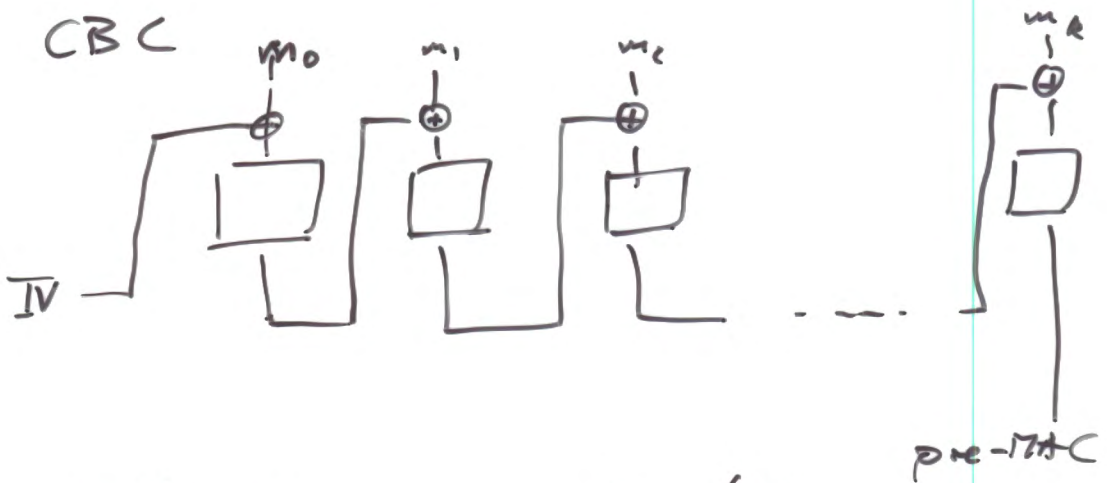
$ctr_i = ctr_{i-1} + 17$ or similar

Advantage: • you may precompute things
→ extremely fast
• provably good if block cipher is good

For authentication we need a
MAC (= message authentication code)
or integrity check scheme
or <u>keyed hash function</u>:

<u>Constructions</u>:

CBC



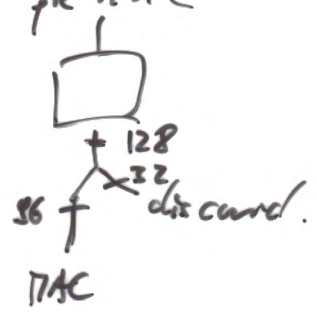
Problem here: appending stuff is not so difficult

$m_0 \dots m_k$ $\quad$ $m_{k+1} \dots m_\ell$

IV ⟶ preMAC$_1$ ⟶ preMAC$_2$

Prevent this by running preMAC
through the block cipher once more.
Possibly cut off some part of the output:

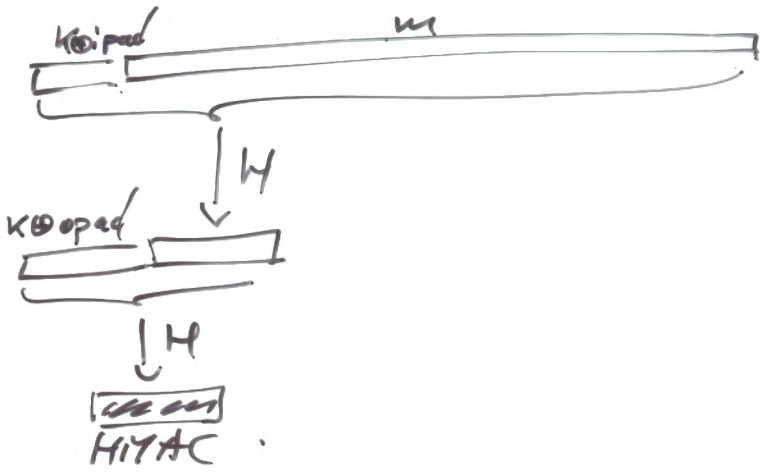
XCBC



12.6.13
cs
(4)

HMAC

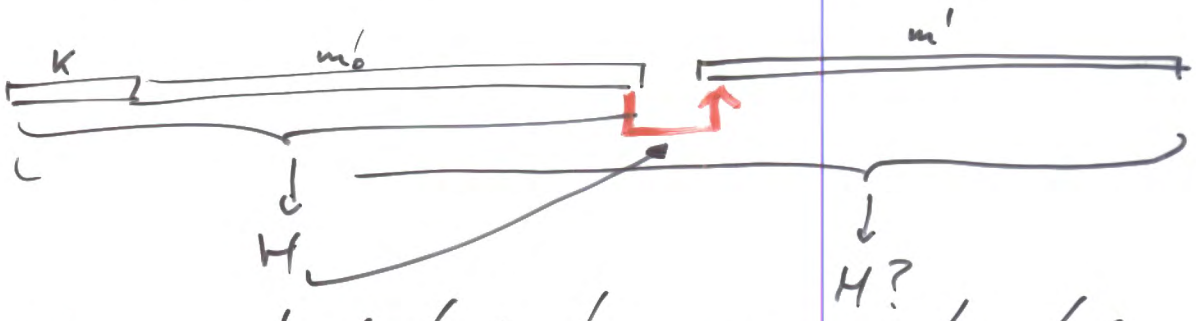
$$HMAC_K^H(m) = H(K \oplus opad \,\|\, H(K \oplus ipad \,\|\, m))$$



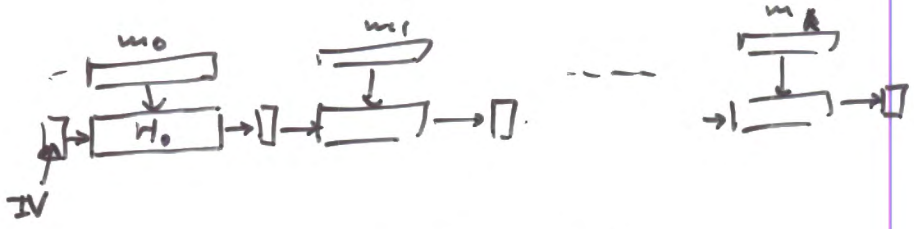
ipad, opad are fixed constants!

(1) Can we omit the key at the end?

No: extension attack.



Fact: most hash functions are constructed from a compression function H_0.

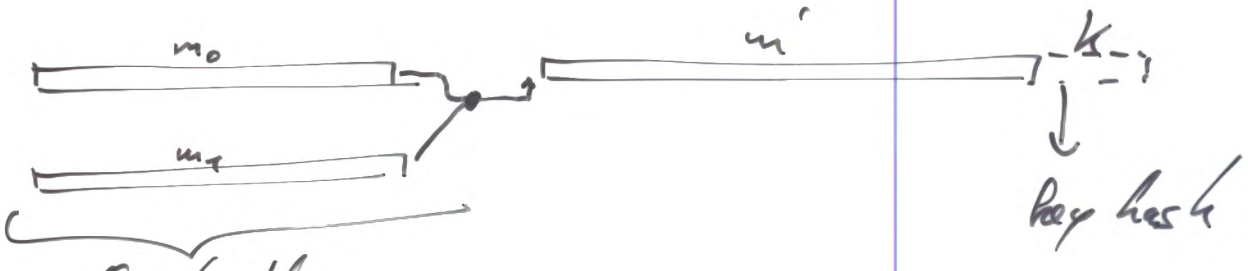


SHA1: message block ~ 512 bits
IV, hash ~ 160 bits.

12.6.13
es
(3)

(2) Can we omit the key at the beginning?

No! assume you have a collision for H



find this offline.

Convince one party to compute the MAC for $m_0 m'$ and you also obtain the MAC for $m_1 m'$.

(3) Why do we use the same key at beginning and end?

⇝ Some kind of meet-in-the-middle attack is possible otherwise.

The HMAC is "almost provably" good.

To be done:

- ETA vs. AtE & NORTON
- pattern of a secure connection.

EtA vs. AtE, HORTON's principle

18.6.13
es
(4)

Should we first authenticate the plaintext and then encrypt or vice versa?

IPsec : EtA.

TLS : EtA ?

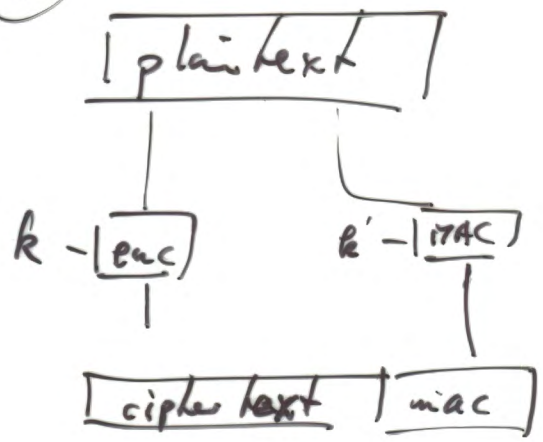
HORTON's principle

A signature or authentication value must depend on the meaning of the plaintext.

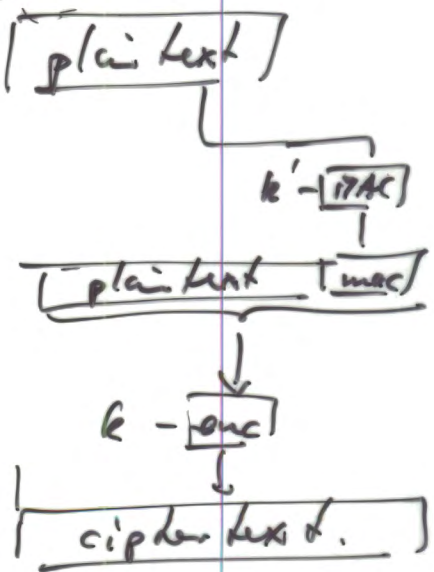
One way to grant that is to authenticate the plaintext itself. → AtE!

AtE

(1a)



(1b)

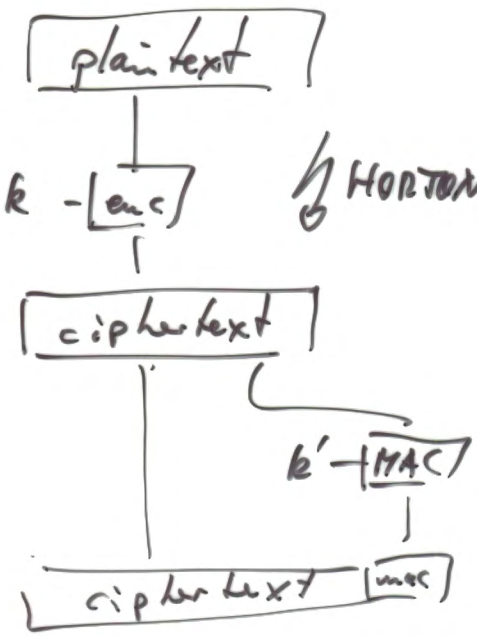


CON: • The recipient has to decrypt every message even he then finds that the mac is invalid.

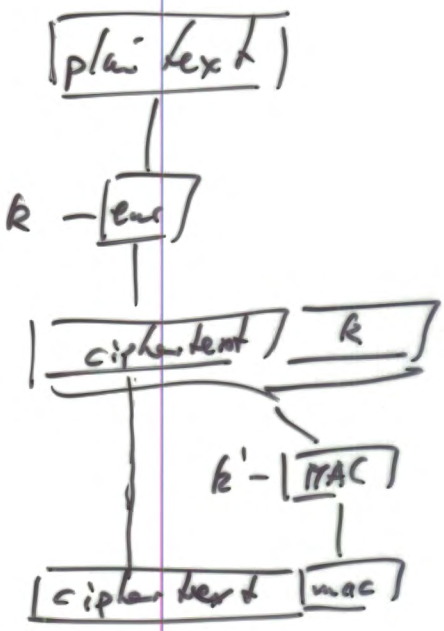
• There might attacks like the ($0 \mapsto 00$, $1 \mapsto \{01, 10$) - stuff ...

EtA

(2a)



(2b)

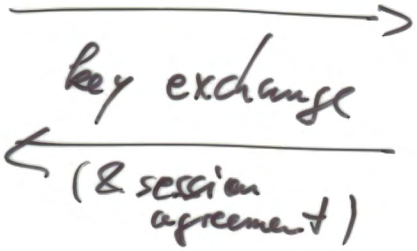


CON: • (2a) does not follow HORTON's principle. If the receiver uses the wrong key k it will consider wrong plaintext as correctly authenticated.

IPsec circumvents this by deriving both k and k' from one source. So they are not independent.

Secure connections

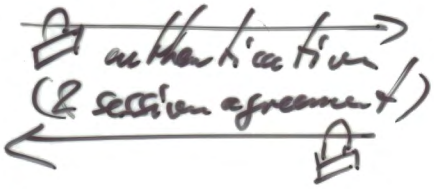
18.6.13
es
(3)



Diffie-Hellman

need:

- a "secure" group
  eg. require that the group size has a large prime factor (or even is a large prime)
- good (pseudo) random numbers generator (& source of truly random bits)
  unpredictable to an attacker.



need:

- public key signatures (or a shared secret key)
- PKI
  → social problems

need:

- FAST algorithms
  - fast encryption
    eg. AES
  - fast authentication + integrity
    eg. HMAC-SHA1
    AES-(X)CBC-MAC

} symmetric key schemes

13.6.18
es
①

e voting

→ Elections?

authentication

participants
who can vote?

free
universal

fair

confidential
(secret)

(~~single~~ one vote / participant
(equal)

direct democracy

counting

possibility to
change a vote

no fraud

candidate
electable

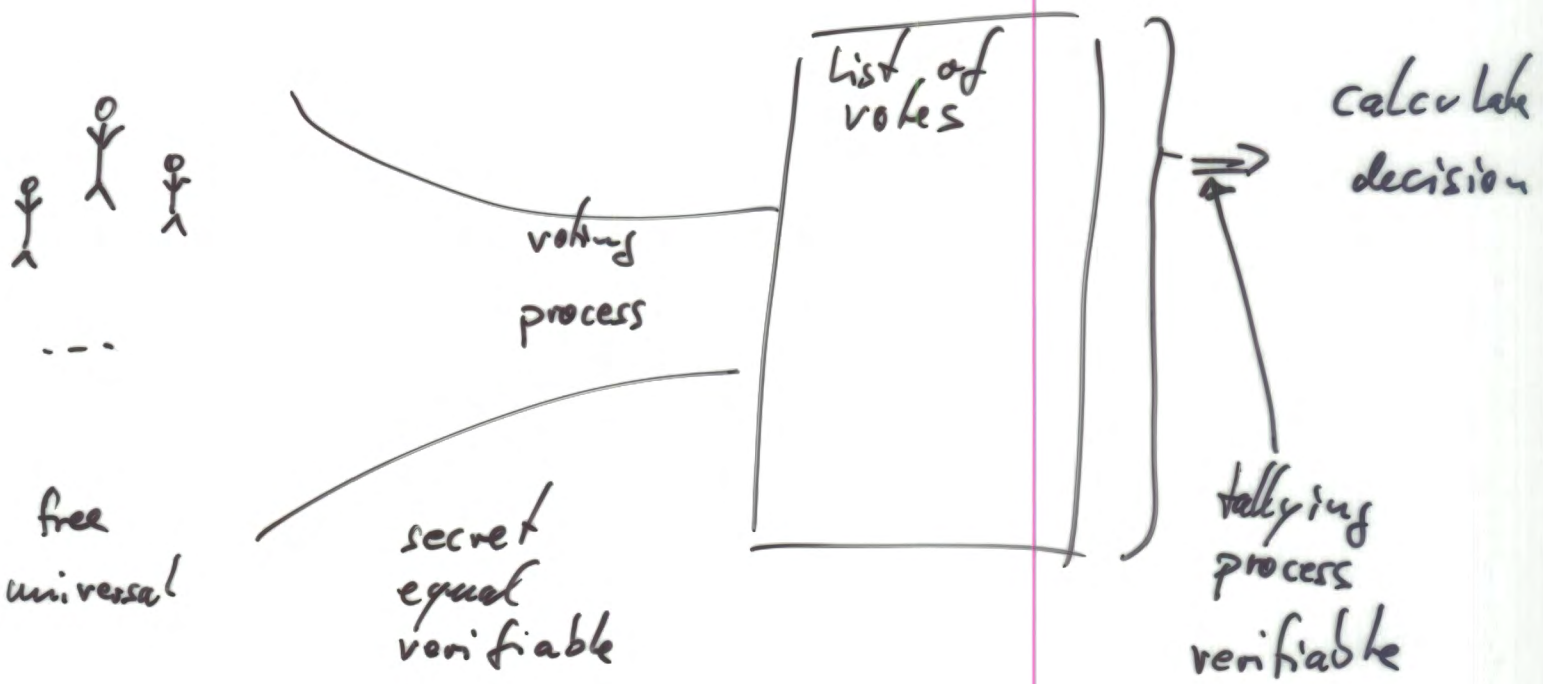
political
party

secure

Democratic elections should be fair & free.!

German constitution requires
free, equal, secret, universal, direct.

The aim of an election is a decision.



Further desirable properties:

- publically verifiable
  - ↳ tallying correct
  - ↳ certain voter is considered.

Voting process:

- Voter goes to a voting office/place.
- Officials check whether the voter is on the list and allowed to vote. They check the identity of the voter before and mark that the voter „has voted".

19.6.13
es
(3)

- The voter gets the ballot, goes to a secret, marks his/her choice, and puts the closed ballot into the voting booth (aka. ballot box).

---

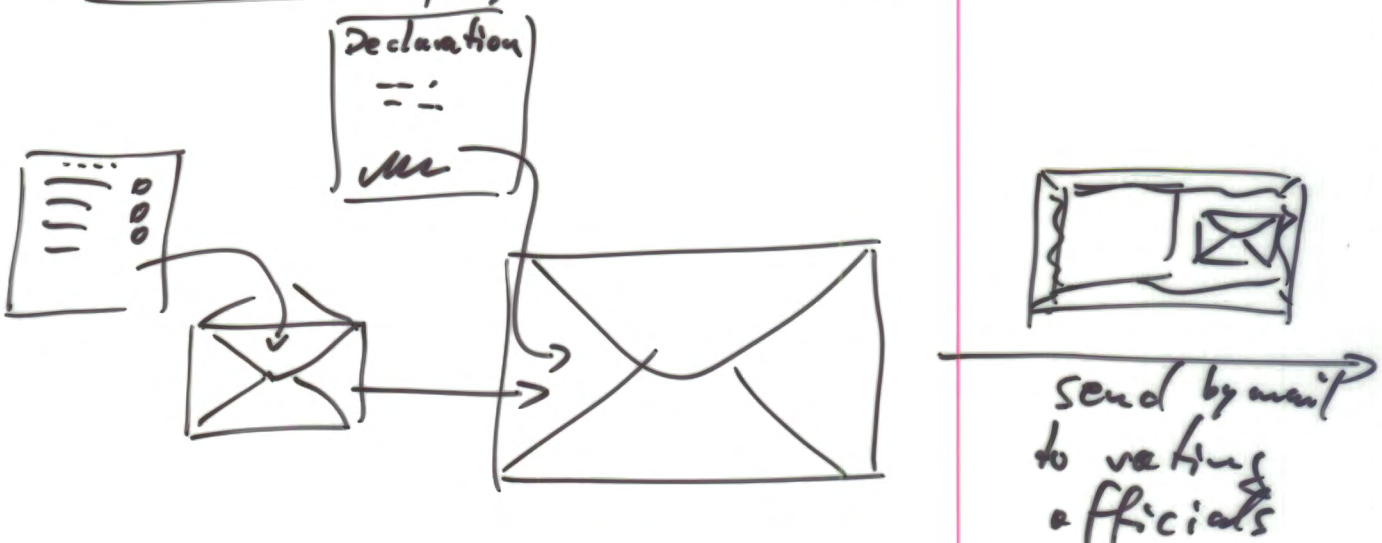
electronic elections

– Forget about voting machines.

We talk about:

electronic voting,
electronic elections,
remote cryptographic elections.

Remote paper voting



25.6.13
es
(1)

Oldest candidate for an electronic election scheme:

Chaum (1981)

Announcement stage (assumed to be complete and authentic).

- System is made public and completely described

  Chaum's decryption mixnet and its RSA public parameters are set up and published.

  } Setting up system and official keys.

- Each voter is associated with a digital signature (key pair).

  } Building list of voters

Registration stage:

(1) Token generation

Each eligible voter V_j generates a random RSA key pair:

K_{V_j} public key

and

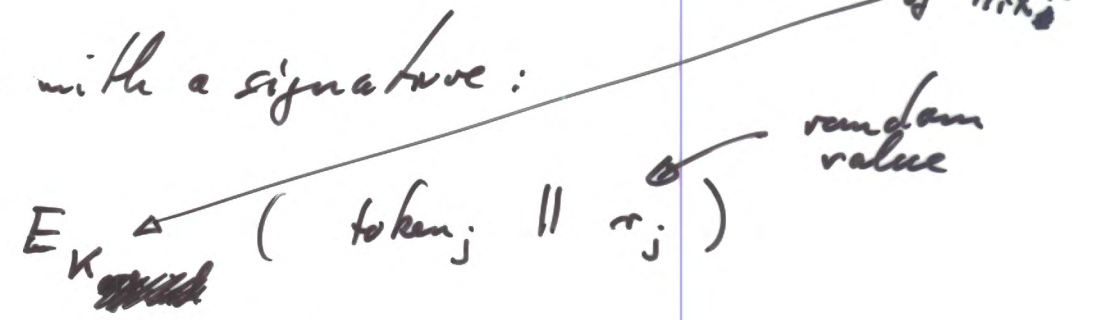
$K_{V_j}^{-1}$ private key.

Let $token_j \leftarrow K_{V_j}$.

(2) The voter V_j sends an encrypted version of his token$_j$ to some official server

$$Mix_1^R$$

together with a signature:



and a signature on this.

The server Mix_1^R checks the signature and whether it corresponds to an eligible voter that has not voted, yet!

If so it sends a receipt to the voter and it sends a partial decryption

$$D_{K_1^R} \left(E_K (\text{token}_j \,||\, r_j) \right)$$

to Mix_2^R. Mix_2^R decrypts again and sends

$$D_{K_2^R} D_{K_1^R} \left(E_K (\text{token}_j \,||\, r_j) \right)$$

to Mix_3^R

This repeats until

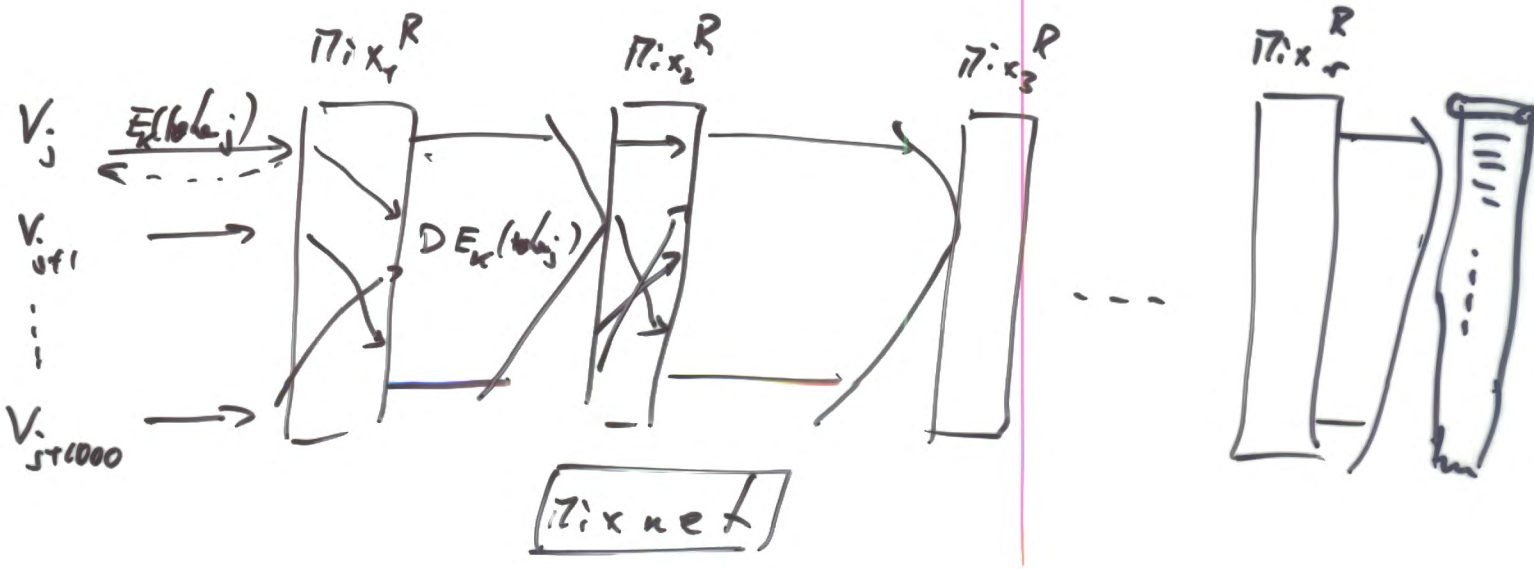
$$D_{K_r}^R \dots D_{K_1}^R \left(E_K (\text{token}_j \| r_j) \right)$$

is output to the list of ~~tokens~~ tokens by Mix_r^R.

To make this work $E_K = E_{K_1}^R \dots E_{K_r}^R$

The last server in the "mixing chain" outputs token_j to the list of authenticated tokens in a sorted fashion.

Actually, each Mix_i^R in the chain bundles many such steps into one message.



25.6.13
es
(4)

Important:

We use randomized encryption in each step!

Without randomization an attacker could easily find

$$D_{K_i^R}^{R} \dots D_{K_1^R}^{R} E_K(token_j)$$

from

$$D_{K_{i+1}^R}^{R} D_{K_i^R}^{R} \dots D_{K_1^R}^{R} E_K(token_j)$$

by simply encrypting with K_{i+1}^R.

In other words: every single encryption step must be randomized:

$$E_K(token_j, r_j)$$

$$= E_{K_1^R}\Big(E_{K_2^R}\big(\dots E_{K_r^R}(token_j \mid r_{j,r}) \mid r_{j,r-1}\big) \dots \mid r_{j,2}\Big) \mid r_{j,1}\Big)$$

The decryption $D_{K_i^R}$ will decrypt and discard the randomness.

This is necessary to ensure that no attacker, even if he compromises all but one of the mixes Mix_i^R, can identify a token of a specific voter.

Actually, the property that we need for every atomic encryption is ~~that~~ IND distinguishability under adaptive Chosen ciphertext attacks: IND-CCA2.

Verification stage

The voter V_j verifies that its token$_j$ arrives on the bulletin board.

We have now achieved that each voter has a key pair and the public of it is on the bulletin board.

Voting stage

The voter V_j encrypts its vote v_j as

$$E_{K_1}^{V}(\ldots\ldots E_{K_e}^{V}(\text{token}_j \,\|\, D_{K_0^{-1}}(v_j \,\|\, 0^k), r_e)\ldots, r_1)$$

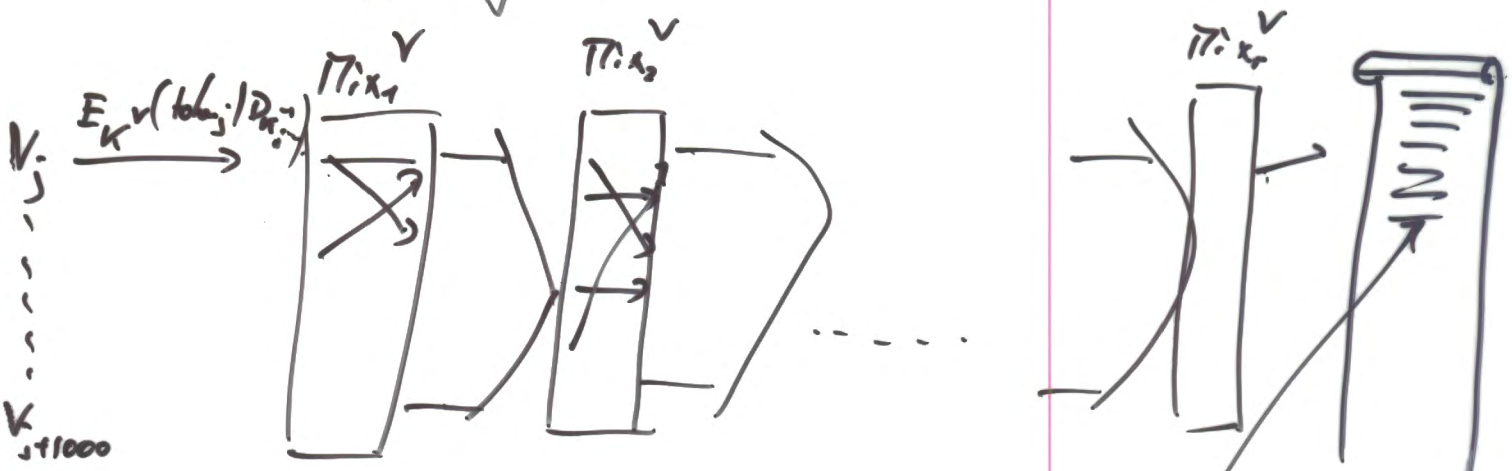
and submits this to voting decryption mix net together with a signature on it.

MIX_1^V checks the signature and id of the voter, decrypts one step, and sends a sorted bundle to next mix.

$\vdots$

$\text{IT.K}_{\text{IT}}^V$ obtains $\text{token}_j \,\|\, D_{V.i^{-1}}(v_j \,\|\, 0^k)$ and

publishes this ~~sorted~~ in a sorted fashion on a second bulletin board as the list of votes.



$\rightarrow v_j \| 0^k$

by encrypting with the token and the format "0^k" ensures that the true private key was used.

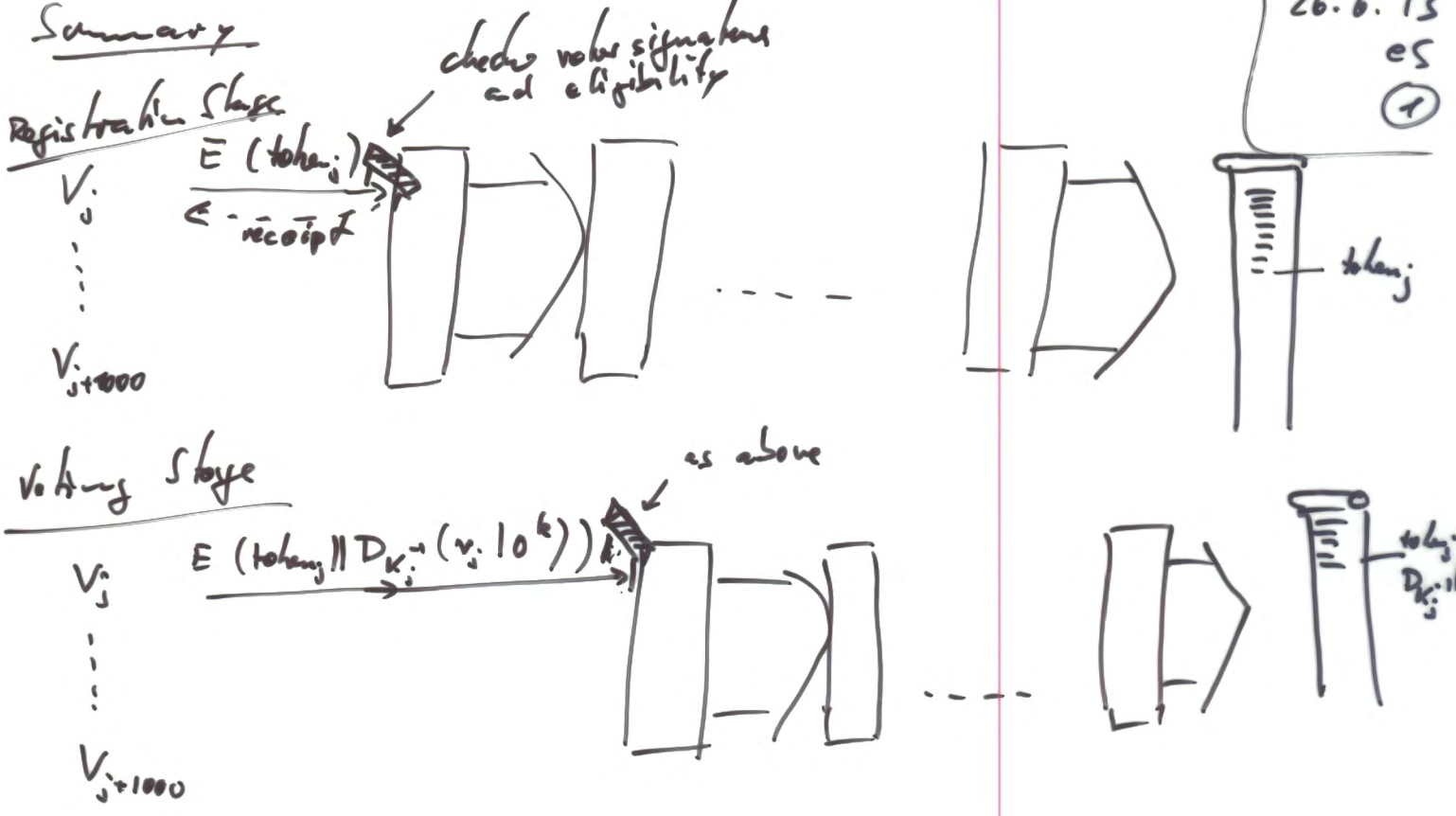
In other words: this is a signed vote.

Tallying stage

Decrypt and count all votes (and derive result).

26.6.13
es
(1)

Summary



Election properties:

general / universal → eligibility

direct → tallying

fair, equal → eligibility, authenticity checks.

free ?

secret ?

The problem is that secrecy can only be kept as long as the voter's private key K_j^{-1} remains uncompromised. But this very key is a receipt for the vote!

The bad guy may force us to reveal this private key!

26.6.13
es
(2)

Eligibility

one man - one vote

Time by authenticity checks.

≤1 : Mix rejects a second try and can prove that it did well. This is based on the voter's signature.

≥1 : ? Need a public list of all signatures handed in to the first mixnet.

Similar in the voting stage. ✓

Anonymity

Assuming that the private key corresponding to the user's token is kept secret, the two mixnet do provide anonymity (see later).

So: PROBLEMATIC.

Verifiability

Individual: Yes, just check whether the voter's token occurs on the second bulletin board.

General:
- Tallying correctness: easy because the votes are on the second bulletin board.
- Eligibility: • verify that each mix outputs as many items as it gets.

26.6.13
es
(3)

- verify that the first mix has accepted at most one vote per voter. Again, the public list of voter signatures is needed.

Summarizing:

we need publically:

- Mixnet keys,
- list of voters including their public signature key.
- list of handed in stuff including voter signatures for Mix_1^P and also for Mix_1^V.
- list of tokens (first bulletin board)
- list of votes (second bulletin board).

Maybe also the intermediate messages sent from Mix to Mix.

This way we can check that the mixnets output the right amount of tokens and votes. We have to rely on each voter doing verification in the verification stage. Otherwise a compromised last mix might replace tokens and votes...

We might add that each voter signs the list of tokens when he verifies it.

(i.e. for the voter)

(→ complicated)

26.6.13
es
(4)

Alternatively, we may have the mixes prove in a non-interactive zero-knowledge proof (or argument) that its input corresponds to its output.

26.6.13
es
(4)

Alternatively, we may have the mixes prove in a non-interactive zero-knowledge proof (or argument) that its input corresponds to its output.

2.7.13
es
(1)

Receipt-freeness

$\not\Rightarrow\not\Leftarrow$ anonymity!

Here: No, unless the user's private key k_i^{-1} stays protected.

Robustness

First problem here: if a voter's token does not arrive on the bulletin board, despite the voter can show his receipt, the recovering is expensive because the entire registration stage (for his precinct) has to be repeated.

In general, we ask ourselves: small problems lead to expensive recoveries...

27.13
es
(2)

Scalability

One needs to make sure that it's still with many voters (millions).

From the basic cryptographic primitives we already have

- open channels
- secure channels

Additionally, the decryption mixnet supplies an

- anonymous channel.

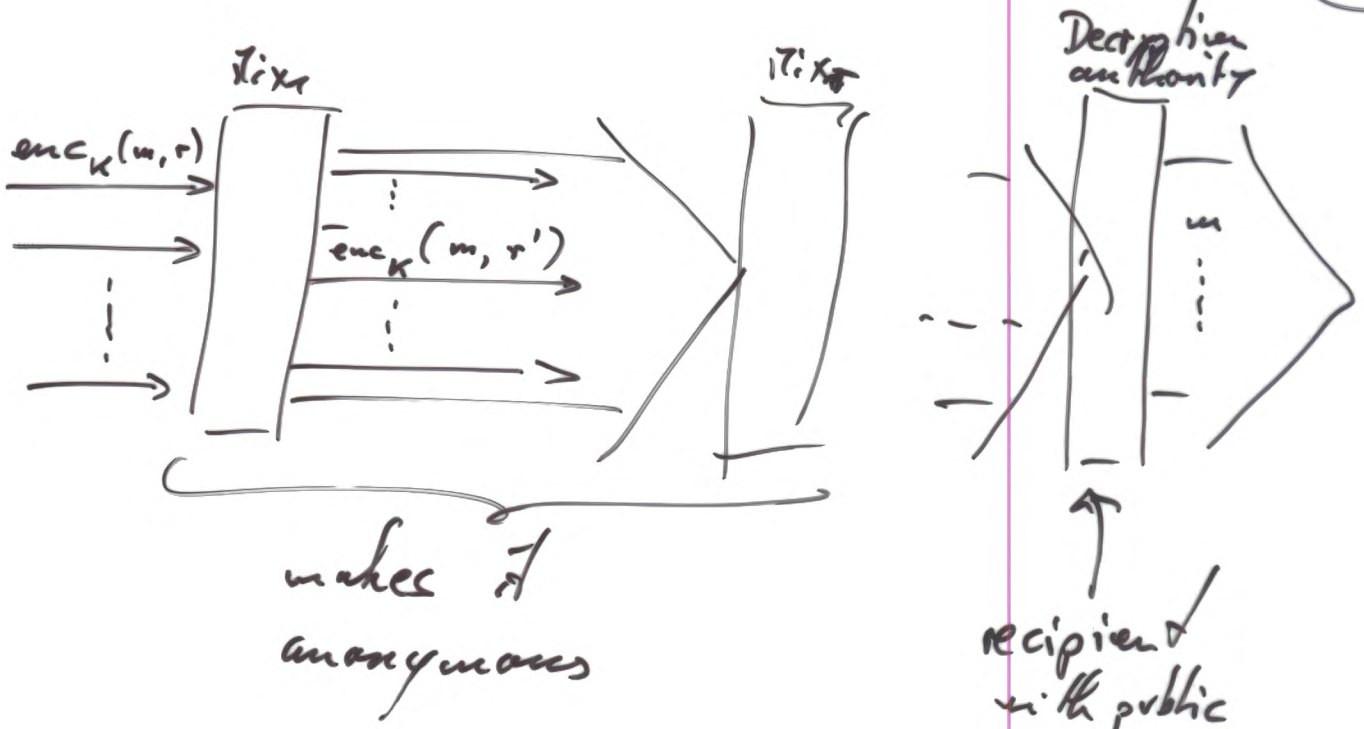
A more efficient version of a mixnet can be obtained by using an encryption scheme which is

- randomized and
- allows reencryption.

Reencryption means that on input of a ciphertext you can construct another ciphertext corresponding to the same plaintext but with another randomness

2.7.13
es
(3)

From this we may construct a "reencryption mixnet":



To do this and more we reconsider ElGamal encryption and gimmicks around it.

ALGORITHM 1.1. El Gamal parameter generation.

Input: Security parameters k, ℓ.

Output: Group G, a prime q, and a generator $P \in G$ of order q.

1. Select a random k-bit prime q.
2. Select an ℓ-bit prime p with $p \equiv_q 1$ and letting $G = \mathbb{Z}_p^\times$ with multiplication. Note that $\#G = p-1$ and by construction $q \mid p-1$.
3. Pick a random element P of order q in G. (Pick an arbitrary random element R of G and consider $P = R^{\frac{\#G}{q}}$. If P is the neutral element of G then retry. Otherwise P has order q.)
4. Return (G, q, P).

ALGORITHM 1.2. El Gamal parameter generation.

Input: Security parameters k.

Output: Group G, a prime q, and a generator $P \in G$ of order q.

1. Select a random k-bit prime p.
2. Repeat 3–8
3. Select a point $P = (x_P, y_P) \xleftarrow{\$} \mathbb{F}_p \times \mathbb{F}_p$.
4. Select a value $a \xleftarrow{\$} \mathbb{F}_p^\times$.
5. Set $b = y_P^2 - (x_P^3 + a x_P)$.
6. If $4a^3 + 27b^2 = 0$ in $\mathbb{F}_p$ then try again.
7. Let G be the elliptic curve given by

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

over $\mathbb{F}_p$. [Its points are all solutions (x, y) of the equation and a further special point $\mathcal{O}$ at infinity. In particular, P is a point.
Addition of two points Q_1 and Q_2 is essentially defined as follows: consider the line through the points and find the third point Q_3 of intersection with the curve. Define $Q_1 + Q_2 := -Q_3$ by mirroring at the x-axis.]

8. Determine $q = \#G$.
9. Until q prime
10. Return (G, q, P).

ALGORITHM 1.3. El Gamal key pair generation.

Input: El Gamal parameters (G, q, P).

Output: A key pair with private key $x \in \mathbb{Z}_q$ and public key $X \in G$.

1. Choose $x \xleftarrow{\$} \mathbb{Z}_q^\times$.
2. Let $X \leftarrow xP$.
3. Return (x,X)

ALGORITHM 1.4. Homomorphic El Gamal encryption.
Publicly known: El Gamal parameters (G, q, P).
Input: The recipient's public key $X \in G$ and the message $M \in G$.
Output: The ciphertext $\mathrm{enc}_X(m)$.

1. Pick a unpredictable temporary private key $t \xleftarrow{\$} \mathbb{Z}_q$.
2. Return $(tP, M + tX)$

ALGORITHM 1.5. Homomorphic El Gamal decryption.
Publicly known: El Gamal parameters (G, q, P).
Input: The recipient's private key $x \in \mathbb{Z}_q$, the ciphertext $(T, Y) \in G \times G$.
Output: The plaintext $\mathrm{dec}_x(T, Y)$.

1. Return $Y - xT$

ALGORITHM 1.6. El Gamal reencryption.

Publicly known: El Gamal parameters (G, q, P).

Input: The recipient's public key $X \in G$ and a ciphertext $(T, Y) \in G \times G$.

Output: A ciphertext $\text{enc}_X(m)$.

1. Pick a unpredictable temporary private key $t' \in \mathbb{Z}_q$.
2. Return $(t'P + T, t'X + Y)$

To decide whether this encryption
a 10 1 . . 1 10 10 . . 1
So this reduces that equivalently we have
to decide whether (T, Y) encrypts 0.

Diffie-Hellman.

3.7.13
es
(1)

Excursion: There exist groups with easy DDH.

[elliptic curves with "small" "embedding degree".]

We want a group with a difficult DDH problem!

Problem: ElGamal is not IND-CCA secure, but only IND-KO or ?-CCA.

Yes → non-malleable ElGamal encryption

[Recall: NM-CCA ≡ IND-CCA.]

3.7.13
ps
(2)

ALGORITHM 2.1. Non-malleable El Gamal encryption.
Publicly known: El Gamal parameters (G, q, P).
Input: The recipient's public key $X \in G$, the message $M \in G$.
Output: The ciphertext $\text{nmenc}_X(m)$.

1. Pick two random temporary keys $t, u \xleftarrow{R} \mathbb{Z}_q$.
2. Encrypt $(T, Y) \leftarrow (tP, M + tX)$.
3. Compute a challenge $c \leftarrow \mathbb{Z}_q(\text{hash}(uP, T, Y)) \in \mathbb{Z}_q$.
4. Compute the response $r \leftarrow u + ct$ in Z_q.
5. Return (T, Y, c, r)

Notice that r depends on t or equivalently T !

ALGORITHM 2.2. Non-malleable El Gamal decryption.
Publicly known: El Gamal parameters (G, q, P).
Input: The recipient's private key $x \in \mathbb{Z}_q$, the ciphertext $(T, Y, c, r) \in G \times G \times \mathbb{Z}_q \times \mathbb{Z}_q$.
Output: The plaintext $\text{nmdec}_x(T, Y, c, r)$.

1. Compute $U \leftarrow rP - cT$ and $c' \leftarrow \mathbb{Z}_q(\text{hash}(U, T, Y)) \in \mathbb{Z}_q$.
2. If $c' \neq c$ then Return Failure
3. Return $Y - xT$

If (T, Y, c, r) comes from Alg. 2.1

then

$$U = rP - cT$$
$$= (v + ct)P - cT$$
$$= vP + c\underbrace{tP}_{T} - cT = vP.$$

Thus $(U, T, Y) = (vP, T, Y)$

and so $c' = \mathbb{Z}_q(\text{hash}(\ldots)) = c$.

3.7.13
es
(4)

An attacker that tries to maulleate
a given ciphertext (T, Y, c, r)
would have to adapt c, r
to the modified $T^* = T + t^* P$
and $Y^* = Y + t^* X$.

But now
$$c^* = \mathbb{Z}_q(\text{hash}(v^* P, T^*, Y^*))$$
where the attacker may use a new v^*
is completely different.
And the attacker does not know the discrete
log of T^* wrt. P and so
he cannot compute
$$r^* = v^* + c^* (t + t^*)$$
... so it looks bad for the attacker.
Actually, the attacker's task is
given T, Y, c, r
find T^*, Y^*, c^*, r^*

such that with $U^* := r^* P - c^* T^*$, $c'^* = \mathbb{Z}_q(\text{hash}(U^*, T^*, Y^*))$
we have $c'^* = c^*$
and $Y - x T = Y^* - x T^*$.

3.7.13
es
(5)

Theorem

ElGamal key generation providing
a group with a difficult DDH
and Algs 2.1 and 2.2
are NM-CCA - secure
or IND-CCA - secure.

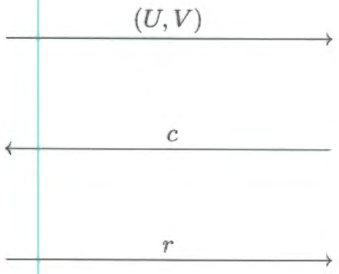
PROTOCOL 3.1. Interactive zero-knowledge proof of equality of discrete logarithms.

Publicly known: El Gamal parameters (G, q, P).

Public input: Group elements $P, T, X, Y \in G$.

Private input to the prover: The discrete logarithm t of T wrt. P and of Y wrt. X, ie. $t \in \mathbb{Z}_q$ such that $T = tP$ and $Y = tX$.

1. The prover chooses a temporary private key $u \xleftarrow{R} \mathbb{Z}_q$ and computes $U \leftarrow uP$ and $V \leftarrow uX$ in G. She sends U and V to the verifier.
2. The verifier chooses a challenge $c \xleftarrow{R} \mathbb{Z}_q$ and sends it to the prover.
3. The prover computes the response $r \leftarrow u + ct$ and sends it to the verifier.
4. The verifier checks that $rP = U + cT$ and $rX = V + cY$.



9.7.13
es
(2)

Thm 3.1 is a zk proof.
↑
honest-verifier

Proof Completeness: Plug in r to verifier's checks:

$$rP = (u+ct)P = uP + c\,tP$$
$$= U + cT.$$

and

$$rX = (u+ct)X = uX + c \cdot tX$$
$$= V + c \cdot Y. \quad \checkmark$$

Soundness: Malicious prover.

Assume Paula can trick Victor for only two possible challenges c_1, c_2.

That is Paula can find (U,V) and r_1, r_2 such that

$$(U,V), c_1, r_1 \text{ is ok}$$

and

$$(U,V), c_2, r_2 \text{ is ok}.$$

But this means

$$\left.\begin{aligned} r_1 P &= U + c_1 T, \\ r_1 X &= V + c_1 Y \end{aligned}\right\} \in G$$

and

$$\left.\begin{aligned} r_2 P &= U + c_2 T, \\ r_2 X &= V + c_2 Y. \end{aligned}\right\}$$

Claim: $=$
Assumption: $\neq$

Say $U = uP$, $V = vP$, $T = tP$, $Y = yX$.

Thus we obtain:

$$\left.\begin{aligned} r_1 &= v + c_1 t_1 \\ r_1 &= v + c_1 y, \\ r_2 &= v + c_2 t_1 \\ r_2 &= v + c_2 y. \end{aligned}\right\} \text{ in } \mathbb{Z}_q$$

Thus

$$\left.\begin{aligned} r_1 - r_2 &= (c_1 - c_2)\, t \\ r_1 - r_2 &= (c_1 - c_2)\, y \end{aligned}\right\} \text{ Note: } c_1 \neq c_2 \text{ ie. } c_1 - c_2 \in \mathbb{Z}_q^{\times}.$$

so

$$t = \frac{r_1 - r_2}{c_1 - c_2} = y.$$

But that contradicts the assumption that the prover is malicious and the claim wrong.

Zero knowledge: Malicious verifier.

Tricky, possibly unprovable.... :(

Honest-verifier zero-knowledge: semi-honest verifier.

We have to define a simulator

SAM

who generates transcripts with the same distributions than actual conversations

but without knowledge of any private input of the prover.

SAM is polynomial-time restricted, the same as VICTOR.

Recall the protocol:

$v \in_R \mathbb{Z}_q$

$U := vP$

$V := vX$ $\xrightarrow{(U,V)}$

$c \in_R \mathbb{Z}_q$

$\xleftarrow{c}$

$r := v + ct$ $\xrightarrow{r}$ Check $rP = U + cT$, $rX = V + cY$.

SAM shall construct a transcript $(U,V), c, r$ without knowing t and with the same distribution.

SAM

1. Pick $c \in_R \mathbb{Z}_q$, $r \in_R \mathbb{Z}_q$.
2. Construct $U, V \in G$ such that holds, i.e. $U = rP - cT$, $V = rX - cY$.
3. return $(U,V), c, r$.

9.7.13 es (4)

The probability of $(U,V), c, r$ solving ~~is a coor~~ occurring in a conversation is

$$\frac{1}{q} \cdot \frac{1}{q} = \frac{1}{q^2}$$

↑ choice of the private $u \in_R \mathbb{Z}_q$; ↑ choice of the challenge $c \in_R \mathbb{Z}_q$

and the prob of the same to occur as an output of SAM is

$$\frac{1}{q} \cdot \frac{1}{q} = \frac{1}{q^2}.$$

↑ choice of $c \in_R \mathbb{Z}_q$; ↑ choice of $r \in_R \mathbb{Z}_q$

ie. the distributions are equal.

$$\text{prob}\left(\langle P(u), V \rangle (P, T, X, Y) = ((U,V), c, r) \right)$$

$$\rightarrow \equiv$$

$$\text{prob}\left(SAM(P, T, X, Y) = ((U,V), c, r) \right)$$

□

What about a general malicious verifier? The problem that he could make c depend on U, V. In that SIM would have construct transcripts with the same property. But he picks c before U, V.

To resort we have to make down the number of challenges to something at most polynomial in our security parameter. Eg. only allow 4 pre-chose challenges. Then SIM can try to predict c, rewind, compute (U, V) and retry if c is not the challenge chosen by the malicious verifier after getting (U, V).

PROTOCOL 3.2. Non-interactive zero-knowledge proof of equalitiy of discrete logarithms.

Publicly known: El Gamal parameters (G, q, P).

Public input: Group elements $P, T, X, Y \in G$.

Private input to the prover: The discrete logarithm t of T wrt. P and of V wrt. U, ie. $t \in \mathbb{Z}_q$ such that $T = tP$ and $Y = tX$.

1. The prover chooses a temporary private key $u \xleftarrow{R} \mathbb{Z}_q$ and computes $U \leftarrow uP$ and $V \leftarrow uX$ in G. She sends U and X to the verifier.
2. The prover computes a challenge $c \leftarrow \mathbb{Z}_q(\text{hash}(T, Y, U, V))$ and sends it to the verifier.
3. The prover computes the response $r \leftarrow u + ct$ and sends it to the verifier.
4. The verifier checks that $rP = U + cT$, $rX = V + cY$ and $c = \mathbb{Z}_q(\text{hash}(T, Y, U, V))$.

(U, V) →

c →

r →

Since the number of challenges is large and we assume that the hash function is pseudo-random, ie. no polynomial time algorithm can see a difference to truly random stuff.

So this has similar properties then the interactive version.

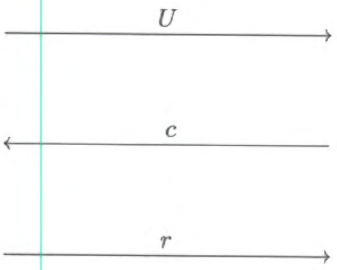
10.7.13
es

PROTOCOL 4.1. Interactive proof of knowledge of a discrete logarithm.

Publicly known: El Gamal parameters (G, q, P).
Public input: Group elements $P, T \in G$.
Private input to the prover: The discrete logarithm of T wrt. P, ie. $t \in \mathbb{Z}_q$ such that $T = tP$.

Claim of the prover: I do know $\frac{T}{P}$.

1. The prover chooses a temporary private key $u \xleftarrow{\$} \mathbb{Z}_q$ and computes $U \leftarrow uP$ in G. She sends U to the verifier.
2. The verifier chooses a challenge $c \xleftarrow{\$} \mathbb{Z}_q$ and sends it to the prover.
3. The prover computes the response $r \leftarrow u + ct$ and sends it to the verifier.
4. The verifier checks that $rP = U + cT$.



Theorem Protocol 4.1 is an interactive, zero-knowledge argument of (honest-verifier)

a proof of knowledge.

Pf Completeness: Ex.

Soundness: malicious but still polynomial time bounded prover.

Ex. & see proof of knowledge prop.

Honest-verifier ZK: Ex. similar to yesterday.

Proof-of-knowledge:

By def. that means that there exists a 'knowledge extractor' which works in place of the verifier but does the power to rewind the prover or to run the prover several times with the same randomness. In our case it's this:

ERNIE

1. Start Paula and obtain U.
2. Send a random challenge c_1 and obtain r_1.
3. Rewind Paula (or restart with same randomness).
4. Send a random challenge $c_2 \neq c_1$ and obtain r_2.
5. Return $\frac{r_2 - r_1}{c_1 - c_2} \in \mathbb{Z}_q$.

10.7.13
cs
(e)

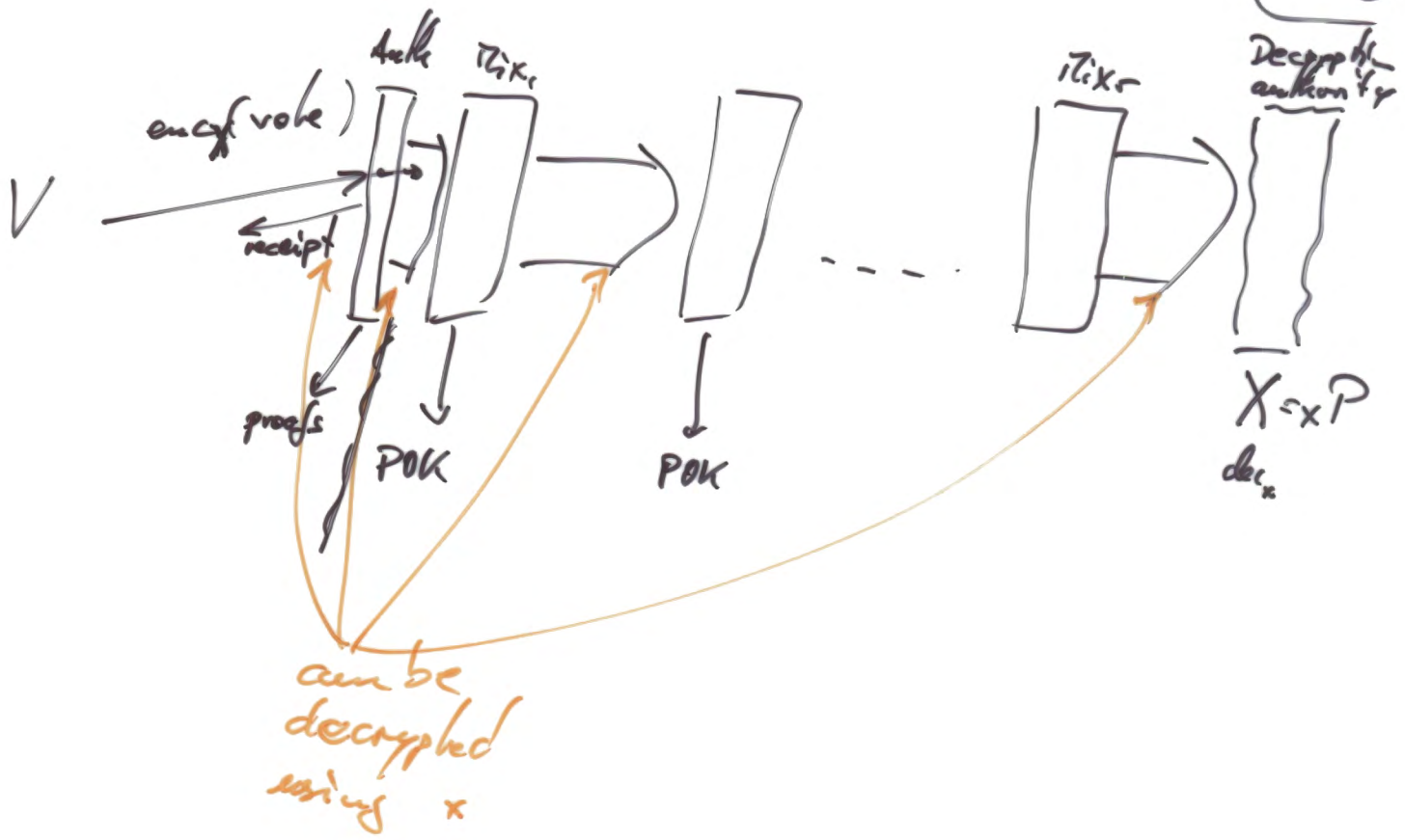
Rewinding makes sure that v is the same in both runs. And by definition of the honest prover we know that

$$r_1 = v + c_1 t$$
$$r_2 = v + c_2 t$$

Since $c_1 \neq c_2$ we may solve this and obtain t from $r_1 - r_2 = (c_1 - c_2) \cdot t$. Thus the above ERNIE always gets the private input of the prover.

This in turn proves that PAULA cannot perform the conversation without the knowledge of t. And so it's a proof of knowledge. □

So we may envision an election scheme:



Idea: Split the decryption authority key to prevent misuse.

→ Basically, there will be parties $1 \dots s$ and each has a private key x_i and together their public key is X:

$$X = x_1 P + x_2 P + \dots + x_s P .$$

Problem: if party 1 would want to cheat it could tell that $xP - x_2 P - \dots - x_s P$ is its public key. Then

$$X = xP .$$

PROTOCOL 5.1. Distributed key generation.

Publicly known: El Gamal parameters (G, q, P).

Input to S_i: Id i and connections to all other share holders S_j.

Private output to S_i: Private key shares x_i.

Output: A public key X, and public key shares X_i.

1. Share holder S_i chooses a private key share $x_i \xleftarrow{R} \mathbb{Z}_q$ and compute $X_i \leftarrow x_i P \in G$.
2. Share holder S_i publishes (ie. sends to all other share holders) a commitment $\text{hash}(X_i)$ on its public key share X_i.
3. Wait until all share holders are done so far.
4. Share holder S_i publishes X_i and proves knowledge of x_i non-interactively, ie. publishes $\text{KnowDlog}(P, X_i)$.
5. Wait until all share holders are done so far.
6. Each share holder checks all commitments and proofs. If something cannot be verified, shout and stop.
7. Return $X = \sum_i X_i$, $(X_i)_i$

10.7.13
es
6

PROTOCOL 5.2. Distributed decryption.

Publicly known: El Gamal parameters (G, q, P).

Input: The ciphertext $(T, Y) \in G \times G$, and the public shares X_i.

Private inputs: Share holder S_i gets its private key share x_i.

Output: $\text{DistDec}_{(x_i)_i}(T, Y)$.

1. Share holder S_i computes and publishes $T_i \leftarrow x_i T$ and proves equality of discrete logarithms of T_i wrt. X_i and T wrt. P, ie. $\text{EqDlog}(P, T, X_i, T_i)$.
2. Wait until all share holders are done so far.
3. Each share holder checks all proofs. If something cannot be verified, shout and stop.
4. Compute $M \leftarrow Y - \sum T_i$.
5. Return M

T_i, EqDlog(...) →

←

Correctness:

$T = \tilde{t} P \quad , \quad Y = M + \tilde{t} X .$

$T_i = x_i T$ (& proves ...)

$M' = Y - \sum T_i$

$= Y - \sum x_i T$

$= Y - (\sum x_i) \tilde{t} P$

$= Y - \tilde{t} \underbrace{\sum x_i P}_{X_i}$

$= M + \tilde{t} X - \tilde{t} X \quad = M$

10.7.13
es
(7)

6. A more sophisticated zero-knowledge proof

The problem in remote elections is that nobody can see whether the voter is under pressure during his voting. So the above zero-knowledge proof is actually too good, as also a coercer will be convinced by such a proof if he is standing "behind" the voter. But we can do better: The following two zero-knowledge proofs prove the statement:

> The El Gamal ciphertexts (T, Y) and (T', Y') encrypt the same message (for the recipient with public key X)
>
> *or*
>
> the prover knows the voter's private key.

This statement can be proved by the party that generated (T, Y) from (T', Y') or it can be proved by the voter. As zero-knowledge proofs are always witness-indistinguishable, a coercer in the role of the verifier cannot tell which of the two forms he sees.

10.7.13 CS (8)

PROTOCOL 6.1. Interactive designated verifier proof.

Publicly known: El Gamal parameters (G, q, P).

Public input: Group elements $T, Y, T', Y' \in G$ and the public key X_{vid} of the voter vid.

Private input to the prover: The reencryption randomness $z \in \mathbb{Z}_q$ such that $T' - T = zP$ and $Y' - Y = zX$.

1. The prover chooses temporary private keys $s, t, w \xleftarrow{\text{RND}} \mathbb{Z}_q$ and computes in G
   - $\widetilde{T} \leftarrow sP$,
   - $\widetilde{Y} \leftarrow sX$ and
   - $\widetilde{V} \leftarrow tP + wX_{\text{vid}}$.

   She sends $\widetilde{T}$, $\widetilde{Y}$ and $\widetilde{V}$ to the verifier. $\xrightarrow{(\widetilde{T}, \widetilde{Y}, \widetilde{V})}$
2. The verifier chooses a challenge $c \xleftarrow{\text{RND}} \mathbb{Z}_q$ and sends it to the prover. $\xleftarrow{c}$
3. The prover computes the response $r \leftarrow s + z(c + w)$ and sends it to the verifier. $\xrightarrow{(r, t, w)}$
4. The verifier computes
   - $\widetilde{T}' \leftarrow rP - (c+t)(T' - T)$,
   - $\widetilde{Y}' \leftarrow rX - (c+t)(Y' - Y)$ and
   - $\widetilde{V}' \leftarrow tP + wX_{\text{vid}}$.

   He checks whether $\widetilde{T}' \stackrel{?}{=} \widetilde{T}$, $\widetilde{Y}' \stackrel{?}{=} \widetilde{Y}$, and $\widetilde{V}' \stackrel{?}{=} \widetilde{V}$.

10.7.13
es
(8)

PROTOCOL 6.2. Interactive fake designated verifier proof.

Publicly known: El Gamal parameters (G, q, P).

Public input: Group elements $T, Y, T', Y' \in G$ and the public key X_{vid} of the voter vid.

Private input to the prover: The ~~verifier~~ voter's private key x_{vid}.

1. The prover chooses the response $r \xleftarrow{\$} \mathbb{Z}_q$ and random values $a, v \xleftarrow{\$} \mathbb{Z}_q$ and computes in G
   - $\widetilde{T} \leftarrow rP - a(T' - T)$,
   - $\widetilde{Y} \leftarrow rX - a(Y' - Y)$ and
   - $\widetilde{V} \leftarrow vP$.

   She sends $\widetilde{T}$, $\widetilde{Y}$ and $\widetilde{V}$ to the verifier. $\xrightarrow{(\widetilde{T}, \widetilde{Y}, \widetilde{V})}$
2. The verifier chooses a challenge $c \xleftarrow{\$} \mathbb{Z}_q$ and sends it to the prover. $\xleftarrow{c}$
3. The prover computes $t \leftarrow a - c$, $w \leftarrow (v - t)x_{\text{vid}}^{-1}$ in $\mathbb{Z}_q$ and sends (r, t, w) to the verifier. $\xrightarrow{(r, t, w)}$
4. The verifier computes
   - $\widetilde{T}' \leftarrow rP - (c + t)(T' - T)$,
   - $\widetilde{Y}' \leftarrow rX - (c + t)(Y' - Y)$ and
   - $\widetilde{V}' \leftarrow tP + wX_{\text{vid}}$.

   He checks whether $\widetilde{T}' \stackrel{?}{=} \widetilde{T}$, $\widetilde{Y}' \stackrel{?}{=} \widetilde{Y}$, and $\widetilde{V}' \stackrel{?}{=} \widetilde{V}$.