Lecture Notes

Esecurity: secure internet & electronic cash

Michael Nüsken

b-it

(Bonn-Aachen International Center
for Information Technology)

Summer 2012

```
Return-Path: <kip@baroxido.cl>
Received: from postfix.iai.uni-bonn.de (localhost [127.0.0.1])
        by mailbox.iai.uni-bonn.de with LMTPA;
        Tue, 03 Apr 2012 08:37:53 +0200
X-Sieve: CMU Sieve 2.4
X-IAI-Env-From: <Kip@baroxido.cl> : [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 BF56F5C401
        for <nuesken@bit.uni-bonn.de>; Tue,  3 Apr 2012 08:37:52 +0200 (MEST)
        (envelope-from Kip@baroxido.cl)
        (envelope-to nuesken@bit.uni-bonn.de) (1)
        (internal use: ta=0, tu=1, te=0, am=-, au=-)
X-IAI-Env-From: <Kip@baroxido.cl> : [127.0.0.1]
Received: from localhost (localhost [127.0.0.1])
        by mandos.iai.uni-bonn.de (Postfix) with ESMTP id C3094484
        for <nuesken@bit.uni-bonn.de>; Tue,  3 Apr 2012 08:37:52 +0200 (MEST)
        (envelope-from Kip@baroxido.cl)
        (envelope-to nuesken@bit.uni-bonn.de) (1)
X-Amavis-Alert: INFECTED, message contains virus:
        winnow.phish.ts.badtele.28.UNOFFICIAL
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 ZCkIejZ7imG5 for <nuesken@bit.uni-bonn.de>;
        Tue,  3 Apr 2012 08:37:48 +0200 (MEST)
X-IAI-Env-From: <Kip@baroxido.cl> : [131.234.142.9]
Received: from mail.uni-paderborn.de (unknown [131.234.142.9])
        (using TLSv1 with cipher AES256-SHA (256/256 bits))
        (No client certificate requested)
        by mandos.iai.uni-bonn.de (Postfix) with ESMTPS id 71D382E
        for <nuesken@bit.uni-bonn.de>; Tue,  3 Apr 2012 08:37:47 +0200 (MEST)
        (envelope-from Kip@baroxido.cl)
        (envelope-to nuesken@bit.uni-bonn.de) (1)
Received: from pova.uni-paderborn.de ([131.234.200.75] helo=localhost.localdomain)
        by mail.uni-paderborn.de with esmtp (Exim 4.72 nylar)
        id 1SExNC-0003X9-Ok; Tue, 03 Apr 2012 08:37:46 +0200
Received: from mail.uni-paderborn.de
        by pova with queue id 28618216-6;
        Tue, 03 Apr 2012 06:37:44 GMT
Received: from Kip@baroxido.cl ( [37.61.17.14])
        by mail.uni-paderborn.de with SMTP id 1SExN9-0006Zp-8Y;
        Tue, 03 Apr 2012 06:37:44 GMT
        (envelope-from Kip@baroxido.cl)
X-Envelope-From: <Kip@baroxido.cl>
Received: from [37.61.17.14] (helo=131.234.142.9)
        by mail.uni-paderborn.de with smtp (Exim 4.72 amazonia)
        id 1SExN9-0006Zp-8Y; Tue, 03 Apr 2012 08:37:44 +0200
Message-ID: <29672_1333435066_1SExN9-0006Zp-8Y_000xonpxq-qhxa1z53aqy-0600003r@beppelorusso.it>
From: "Rubin Hadley" <Rubin.Hadley@athom.be>
Subject: [SPAM!] Re: 30 days to a new Bachelor, Master or PhD
Date: Tue, 03 Apr 2012 12:31:32 +0500
To: norbert.olivier@upb.de
MIME-Version: 1.0
Content-Type: text/plain; charset=windows-1252
Content-Transfer-Encoding: 7bit
X-PMX-Version: 6.0.0.2142326, Antispam-Engine: 2.7.2.2107409, Antispam-Data: 2012.4.3.63015
X-Perlmx-Spam: Gauge=XXXXXXXXXIIII, Probability=94%, Report='
 RDNS_SUSP_KNOWN_PHONE 5, SXL_IP_DYNAMIC 3, KNOWN_PHONE_EN_AGG 2.5, HTML_00_01 0.05, HTML_00_1
0 0.05, SUPERLONG_LINE 0.05, BODYTEXTP_SIZE_3000_LESS 0, BODY_SIZE_1000_1099 0, BODY_SIZE_2000
_LESS 0, BODY_SIZE_5000_LESS 0, BODY_SIZE_7000_LESS 0, NO_URI_FOUND 0, RDNS_NXDOMAIN 0, RDNS_S
USP 0, RDNS_SUSP_GENERIC 0, __BOUNCE_CHALLENGE_SUBJ 0, __BOUNCE_NDR_SUBJ_EXEMPT 0, __C230066_P
1 0, __C230066_P1_3 0, __C230066_P1_4 0, __C230066_P1_5 0, __C230066_P1_A 0, __C230066_P2 0, __
_C230066_P5 0, __CT 0, __CTE 0, __CT_TEXT_PLAIN 0, __HAS_MSGID 0, __MIME_TEXT_ONLY 0, __MIME_V
ERSION 0, __SANE_MSGID 0, __SUBJ_ALPHA_END 0, __TO_MALFORMED_2 0, __TO_NO_NAME 0'
X-IMT-Spam-Score: 9.4 (+++++++++)
X-IMT-Authenticated-Sender:
X-Spam-Level: ********* at mandos.iai.uni-bonn.de
X-Spam-Score: 9.6 at mandos.iai.uni-bonn.de
X-Spam-Status: No, score=9.561 tagged_above=9999.9 required=9999.9
        tests=[FM_SCHOOLING=2.386, FM_SCHOOL_DIPLOMA=0.776, GET_MORE=0.1,
        IAI_10288a15=0, IAI_10308a15=0, IAI_10341a15=0,
        RCVD_HELO_IP_MISMATCH=3.2, RCVD_NUMERIC_HELO=2, RDNS_NONE=1.1,
        SPF_PASS=-0.001]
X-Spam-Report: namo@mandos.iai.uni-bonn.de pronounced judgment in matters of
        spam: Final score: 9.6 points
        pts rule ------------description----------------------------------------
        -0.0 SPF_PASS               SPF: sender matches SPF record
        3.2 RCVD_HELO_IP_MISMATCH  Received: HELO and IP do not match, but should
        2.0 RCVD_NUMERIC_HELO      Received: contains an IP address used for HELO
```

```
        0.1 GET_MORE                BODY: GET_MORE
        0.0 IAI_10288a15            IAI_10288a15
        1.1 RDNS_NONE               Delivered to trusted network by a host with no rDNS
        0.0 IAI_10341a15            IAI_10341a15
        2.4 FM_SCHOOLING            Meta Combo Phrase for Schooling (2)
        0.0 IAI_10308a15            IAI_10308a15
        0.8 FM_SCHOOL_DIPLOMA       Meta for Schooling + Diploma.
        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
```

Blank Line

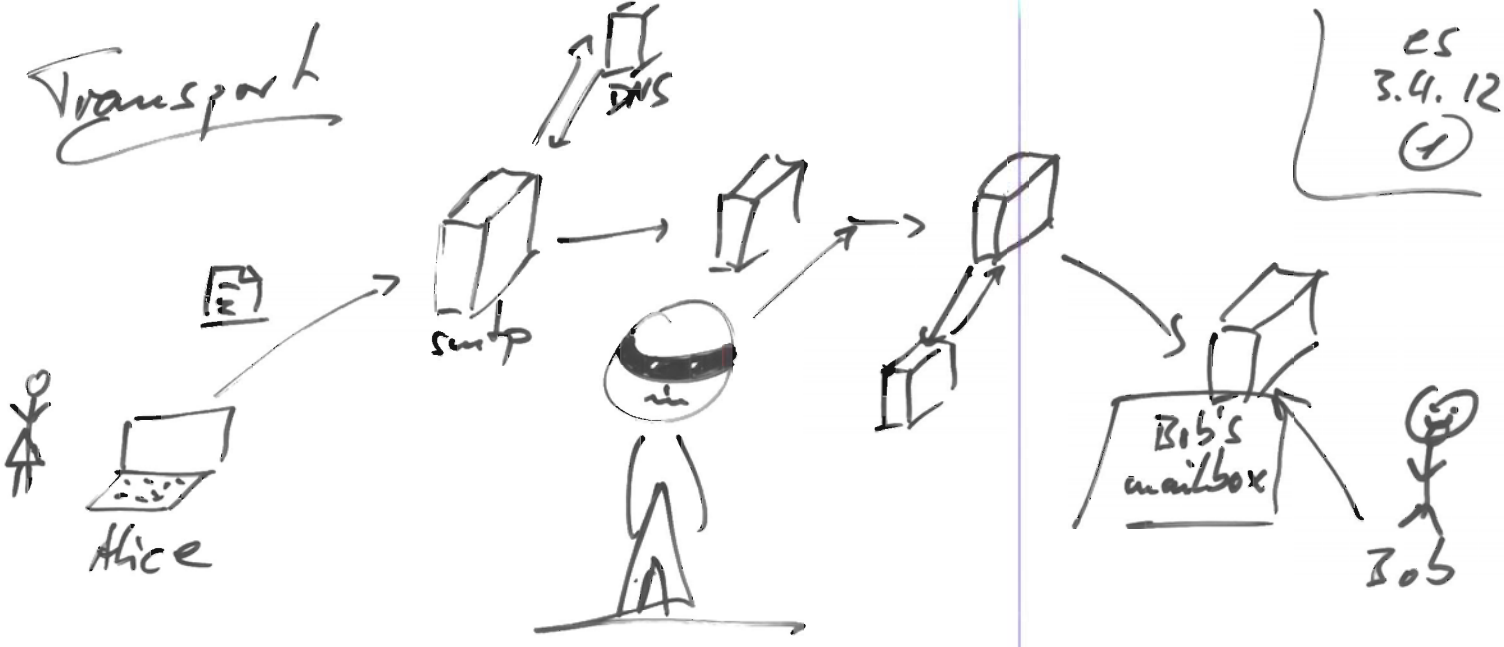
Content Ready

How are you? Norbert.olivier
Your new resume will increase your salary several times.Confirm your real abilities and knowle
dge with the diploma. It is your Chance
Bachelor, Master, MBA, and Doctorate diplomas available in the field of your choice that's rig
ht, you can even become a Doctor and receive all the benefits that comes with it!
Many people share the same frustration, they are doing the work of the person that has the deg
ree and the person that has the degree is getting all the money.

Dial in USA 1-603-509-2001 and outside USA.: +1-603-509-2001 and get more information about de
gree of your dreams! You don\222t need to study or worry about passing those boring exams! You
 don\222t need to attend the lectures! Do you like this idea? Want to know more details? Then
contact us or leave a voicemail with your NAME and contact TELEPHONE NUMBER. (with country-cod
e)
It is not necessary to detain progress of your life. Make a call and we will begin work. We wo
rk 24 hours a day for our clients.

es
3.4.12
(1)

Transport



Security?

→ What do we have?

→ What do we want?

→ History and design?

4.4.12

What is email? What does it do?
What were the design goals?

- provide communication
- simple
- decentralized
- fast
- ~~secure~~
- interoperability
- text only !
- cheap
- anyone can be sender or receiver
- want to receive any mail !

first mails: late 1971

- include address information about sender and receiver.
- relay / forward email
- ease of use

es 4.4.k (1)

→ DNS supplies information about the topology

? Security ?

→ SMTP Simple Mail Transfer Protocol

es
4.4.18
(2)

What do we want?:

Security goals

- keep the content private : confidentiality.
  ⇓
  encryption!
- integrity : nobody can change the message.
  ⇓
  signature
- authenticity : the sender is who we think it is
- identify sender
- identify recipient
- reliability : • message eventually reaches the recipient (correct)
  • service is available at all times
  • ...

safety

DoS

- sender cannot deny the content:
  ~~undeniability~~
  non-repudiation

- proof of delivery
  of submission
  of reaching

One solution to implement that with email: GnuPG

- anonymity of the sender
  ... but still ~~auth~~ prove authorization

4.4.12 (3) es

10.4.12 es (1)

Attack (vs. Email)		Defence
• Intercept emails	→	Encryption
• Spoofing (pretend to be another sender)	→	Signature
• Phishing, social engineering		Caution, knowledge
• Changing the message	→	Signature
• DNS spoofing	...	Encryption, Confirmation request, DNSsec
• Send malware	...	Antivirus software. Sender trust + signature
• SPAM	...	Filters, grey listing, ...
• Denial of Service (DoS)		

CRYPTO

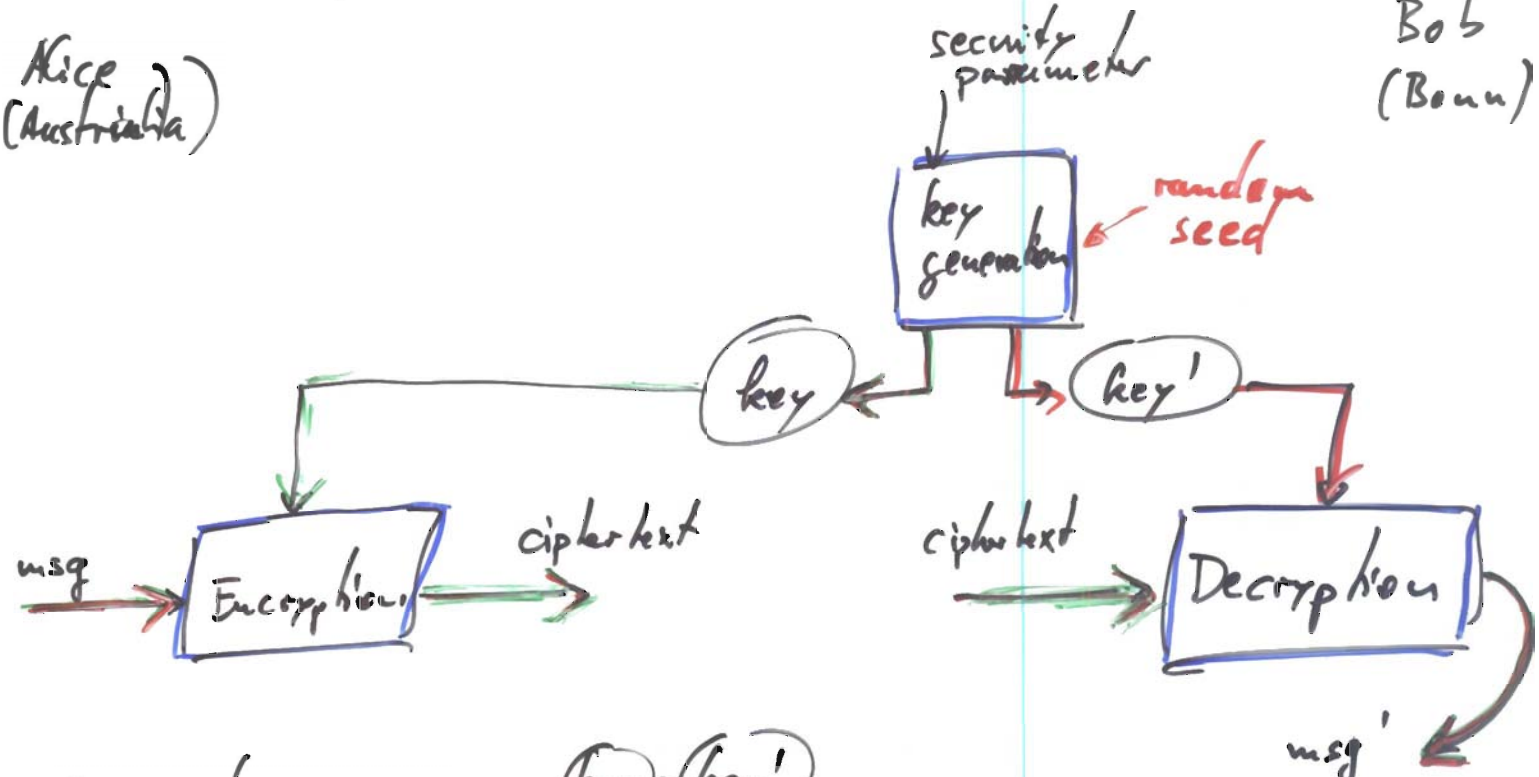
EDUCATION

SAFETY

10.4.12
es
(2)

Our toolbox: cryptography & related

(1) Encryption



Symmetric case: key = key'

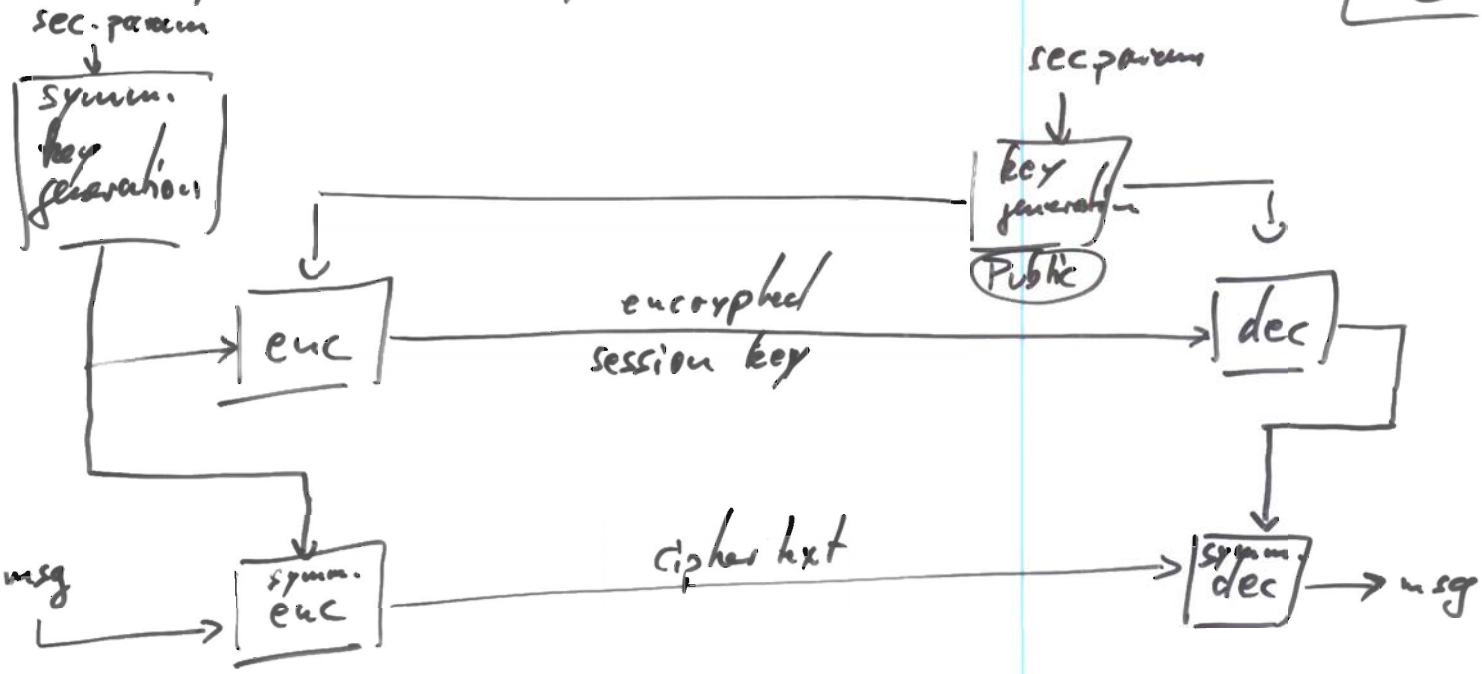
Public-key case: key, key' are somehow related (to ensure that msg' = msg.) such that it's difficult to compute key' from key.

Pros/Cons for public key:

- slower
+ need no prior keys, can use insecure channels
+ need much fewer keys.

10.4.12
es 3

Hybrid encryption unites both and provides all pros!



- Encryption protects against disclosure and grants confidentiality.

Only the owner of key' can decrypt.

! No protection against changes!

Kerckhoffs' principle

The only thing unknown to the attacker is the key.

... or rather the random seed! [entropy!]

Debian bug

11.4.12
es
(4)

Repeat!

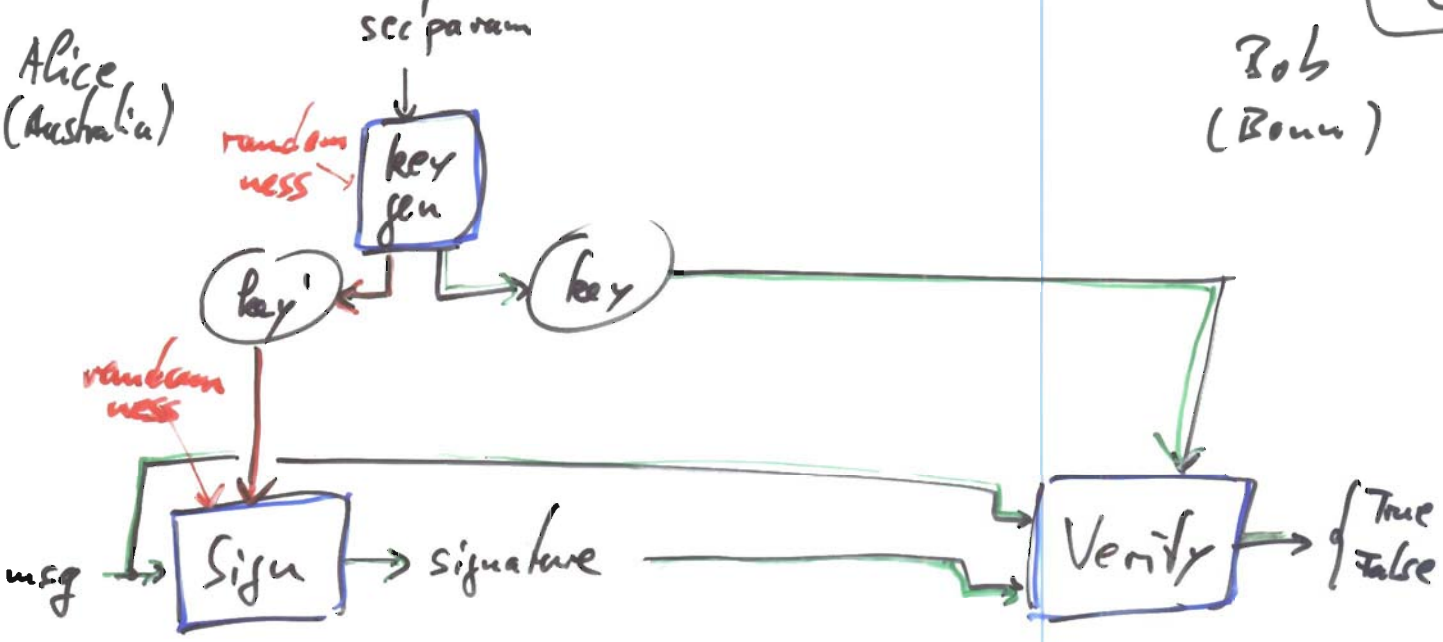
We need good source of true randomness.
There must be enough bits that the attacker cannot predict to prevent him from trying all possibilities.

Secrets are whatever remains unpredictable to an attacker.

Debian bug:
entropy was only ~16.5 bits.
Need ≥ 80 bit...

Large scale internet computations achieve $\approx 2^{60}$ operations.

(2) # Signatures



Correctness: Verify always answers True on (msg, signature) pairs generated within the scheme.
Verify usually (with high probability) answers False for random (msg, signature) pairs.

Efficiency: obvious! all algorithms should be fast (fast means at least poly-time).

Security: ?

Signatures protect against changes to the message, false senders, denial of having written the message.

and provides integrity, identification/authentication, non-repudiation.

Public key scenario: key, key' are different and it is difficult to derive key' from key.

Symmetric key scenario: key = key'

11.4.12
es
(3)

Major difference: in the symmetric scenario the non-repudiation only holds ~~partly~~ partially. Only sender and receiver may generate signatures, but we cannot say ~~how~~ who of the two did it unless we are one one of sender/receiver. A third may not make the distinction.

Side remark: in trial the shared secret may be revealed for verification (after revealing the (msg, signature) pairs).

In the symmetric scenario we call that message authentication code (MAC)

11.4.12
es
(4)

(3) Public key infrastructures,
web of trust,
identity based mechanisms

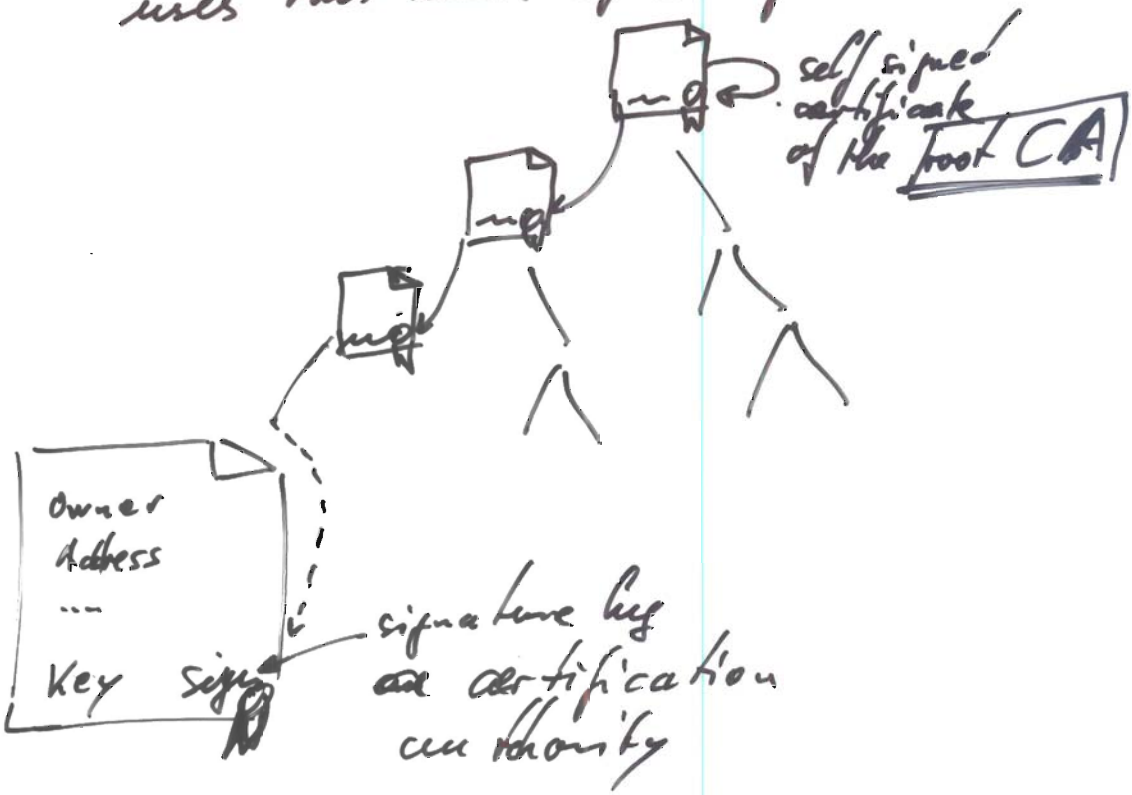
How to trust a public key?

17.4.12
(1)

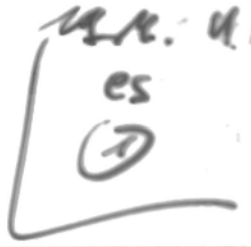
Option 1: Use certificates in a PKI (public key infrastructure).

Eg. TLS/SSL used in https uses this kind of certification.

Idea:



Option 2: In a web of trust ~~we at~~ users sign each other keys provided the know each other and verified the other's identity.



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

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

- Erweiterter Schlüsselgebrauch
- Zertifikatsgegenstand-Schlüssel-ID
- Zertifizierungsstellen-Schüsselidentifikator
- Zertifikatsgegenstand-Alternativ-Name
- CRL-Verteilungspunkte
- Zertifizierungsstellen-Informations-Zugriff
- Zertifikatsunterzeichnungs-Algorithmus
- Signaturwert des Zertifikats

Feld-Wert

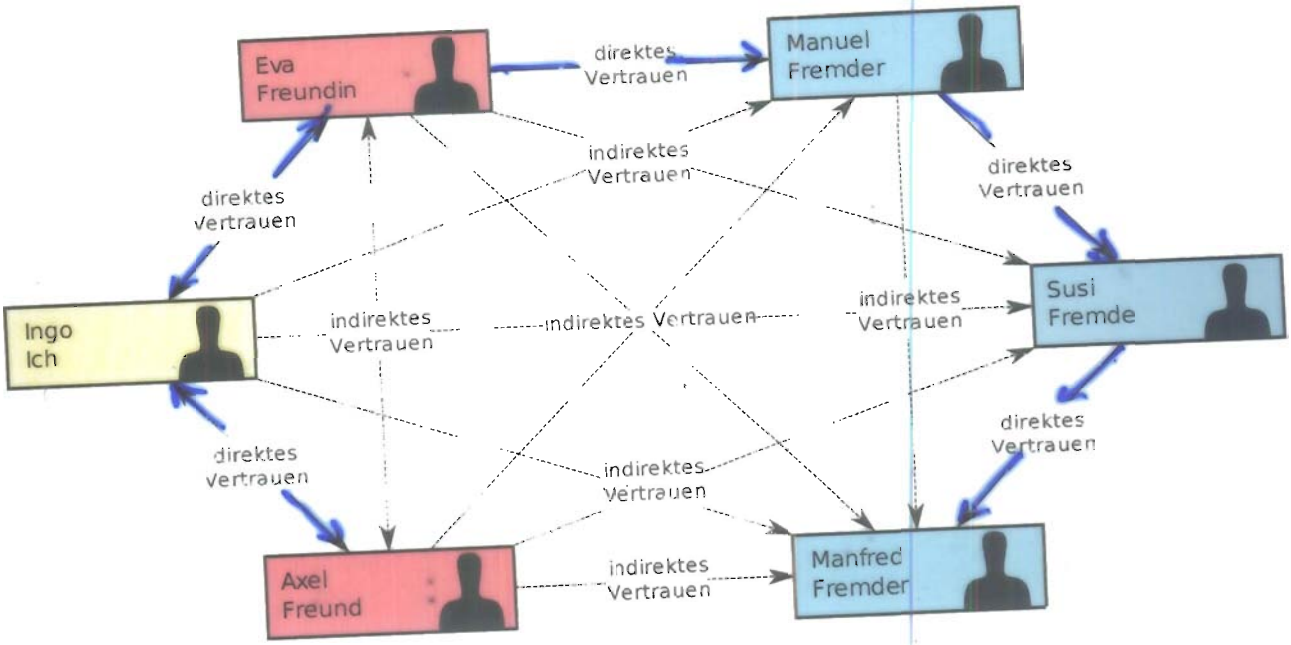
```
Größe: 256 Bytes / 2048 Bits
5d fc 4a ee 80 8f 4f 76 0f 38 33 a7 ce 59 c9 ad
5b 6d ac 20 b7 2d d5 90 aa 2f 9e 3a b3 48 8c 6a
4b fa 81 46 51 7a 37 a0 fd b6 dc 33 86 4f 8e 6f
90 64 6a 06 d1 76 95 1c c0 c4 6e 36 d8 10 4f 4b
72 3d 4b 23 80 83 2f ce 16 99 83 02 9c c3 ea 84
24 e9 d7 ab 55 76 fd 84 57 41 f5 e4 69 a2 c1 11
9f 21 4e 18 e0 11 d9 31 ec 37 93 d8 c2 3d d6 a2
3b 9b 2b 87 a8 70 48 2c 3f 80 3c e6 86 46 c2 97
```

Exportieren...

Schließen

17.4.17
(2)

http://upload.wikimedia.org/wikipedia/commons/4/4e/Web_of_Trust.svg



Here, everybody signs keys that she can verify and thus attributes some trust to those keys.

If you get a signed key you may the list of signatures until you reach someone that you do trust yourself.

This is what PGP and GnuPG use.

Additionally, there are key servers collecting signed keys/certificates.

Option 3: Use Identity Based Schemes, or so-called Certificateless Schemes.

17.4.12
es
(3)

The security of a PKI lies in

- the care of the CA to keep their private keys uncompromised. human factor!
- the care of the CAs in checking the identities before ~~issue~~ issuing a certificate. human factor!
- the identity of the Root CA (which is usually embedded in OSs or Browser or ...) human factor!
- the security of the used signature schemes.
- the security of your computing platform. human factor!

What to look at for a new cryptographic scheme?

- Correctness:
  the scheme fulfills the wanted task if everything works as specified.
- Efficiency:
  all parts of your scheme should be reachably fast, i.e.
  - polynomial-time from a theoretical perspective,
  within
  - concrete time constraints from a practical perspective.
    (eg. 1s or 1 min).

Security

A system is insecure if we can present a successful attack.

What is a successful attack?

(1) It should be efficient (fast), i.e. poly-time or below a certain threshold.

(2) The probability of success should be significant for at least non-negligible $\frac{1}{}$, i.e. $\exists$ poly: prob(A successful) $\geq$ prob(Guessing successful) $+ \frac{1}{\text{poly}(k)}$

Note: (1) + (2) implies that amplification of A leads to an attack with expected poly-time.

[Ex: Expected time of a loop with exit prob. p.
Ex? amplification]

(Actually, prob(A successful) $\leq$ prob(Guessing) $- \frac{1}{\text{poly}(k)}$ may also be good!)

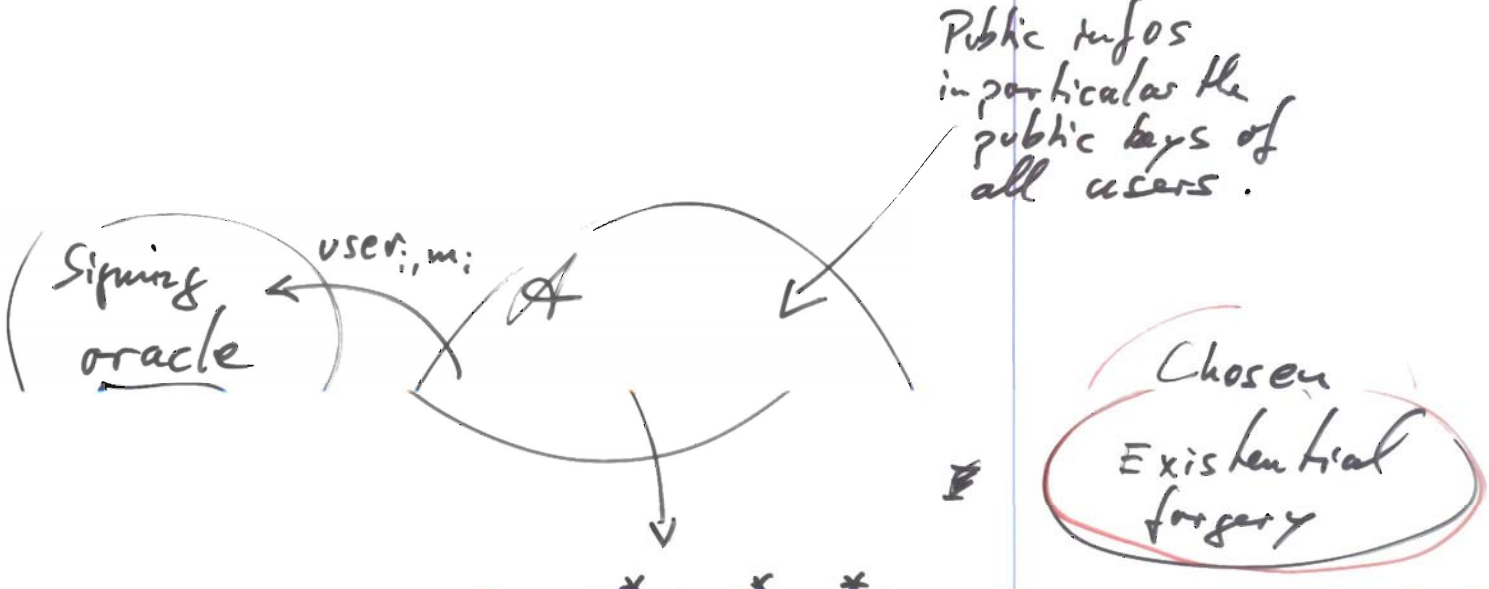
We define the advantage of an attacker:

$\text{adv}_A = |\text{prob}(A \text{ successful}) - \text{prob}(\text{Guessing})|$

(3) We have to formulate what "successful" means.

This – as the correctness requirement – depends on the analyzed scheme.

Example Public-key signatures.



(advantage.

For security we require that no such attacker has poly-time & inv-poly

Example

RSA-signature:

$$\text{Verify}_{(N,e)}(m, \sigma) = \left(\sigma^e \overset{?}{=} m \text{ in } \mathbb{Z}_N\right)$$

To make it a correct scheme we have to define

$$\text{Sign}_{(N,d)}(m) = m^d \text{ in } \mathbb{Z}_N.$$

Correctness ✓

Efficiency ✓

Security?

Is this scheme EUF-CMA secure?

No: It does this:

1. Choose σ at random.
2. Compute $m \leftarrow \sigma^e$.
3. Return (m, σ).

Example RSA-FDH

Assume $h: \{0,1\}^* \to \mathbb{Z}_N$ is a cryptographic hash function (collision-resistant and one-way) and (more or less) surjective.

Now let

$$\text{Verify}_{(N,e)}(m, \sigma) = \left(\sigma^e = h(m) \text{ in } \mathbb{Z}_N\right).$$

For correctness we need

$$\text{Sign}_{(N,d)}(m) = h(m)^d \text{ in } \mathbb{Z}_N.$$

It's better!

Actually: if RSA-FDH with some h is EUF-CMA secure then h is one-way.

18.4.12 (5)

Proof Assume h is not one-way.
Then the f. attacher is successful, fast, ...:

1. Pick $\sigma \in \mathbb{Z}_N$ at random.
2. Compute σ^e.
3. Find a message m such that $h(m) = \sigma^e$. ← one-way attacher
4. Return (m, σ).

This attackers is only slightly larger than the run-time of the one-way attacher, thus poly-time.
And its advantage is equal to the advantage of the one way attacher, thus non-negligible.
Thus RSA-FDH with this h cannot be EUF-CMA secure. □

24.4.12 (1)

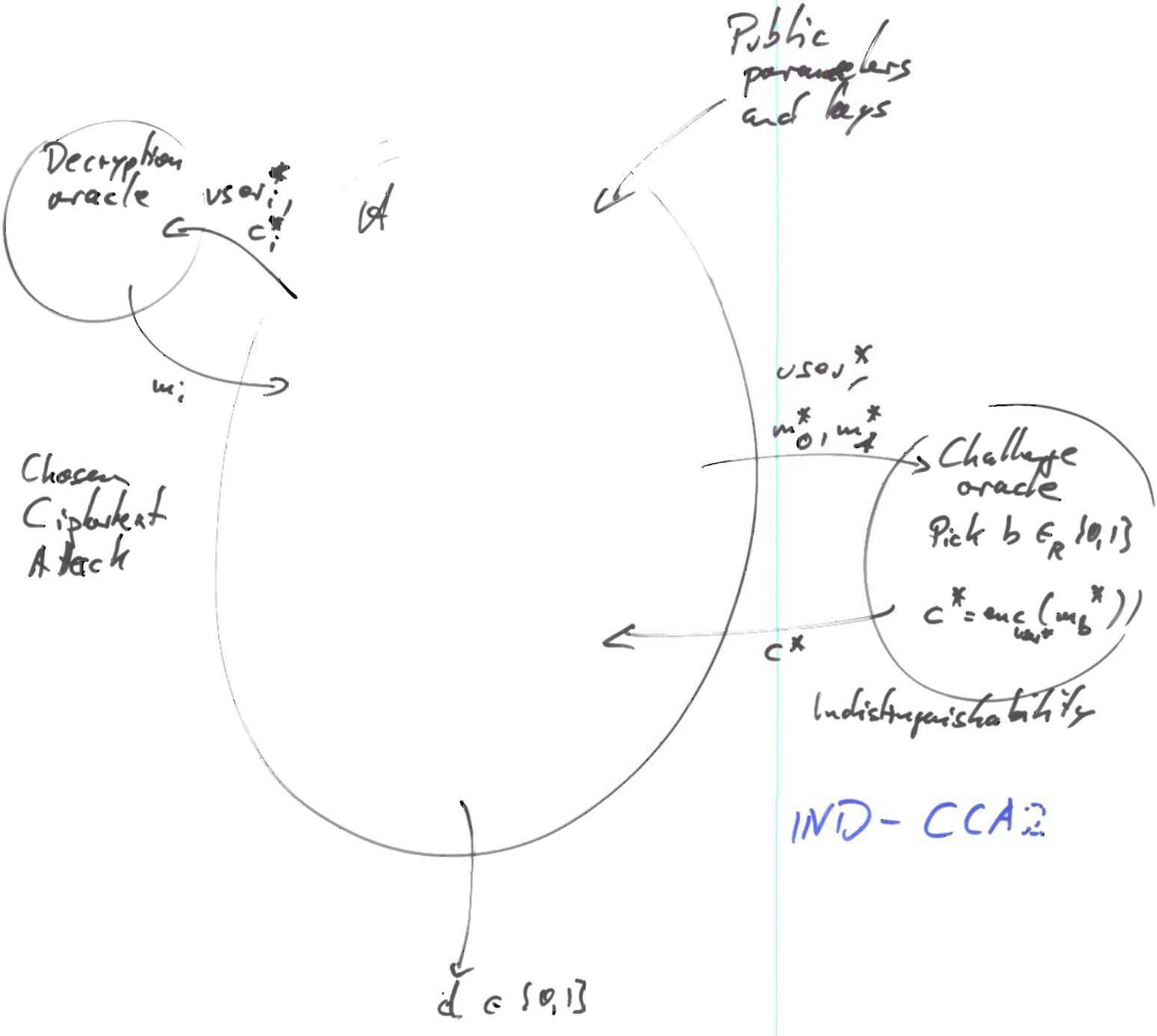
Theorem RSA-FDH is EUF-CMA secure in the Random Oracle Model.

Note Security in the ROM $\not\Rightarrow$ Security in the standard model

$\exists h \forall \ldots$

$\forall h \exists \ldots$

Example Public key encryption schemes



$\mathcal{A}$ successful $:\Leftrightarrow$ $d = b$

$\wedge \quad c^* \notin \{c_1, \dots\}$

A scheme is IND-CCA secure if no efficient attacker with non-negligible advantage exists.

Good news: NMA-CCA $\Leftrightarrow$ IND-CCA

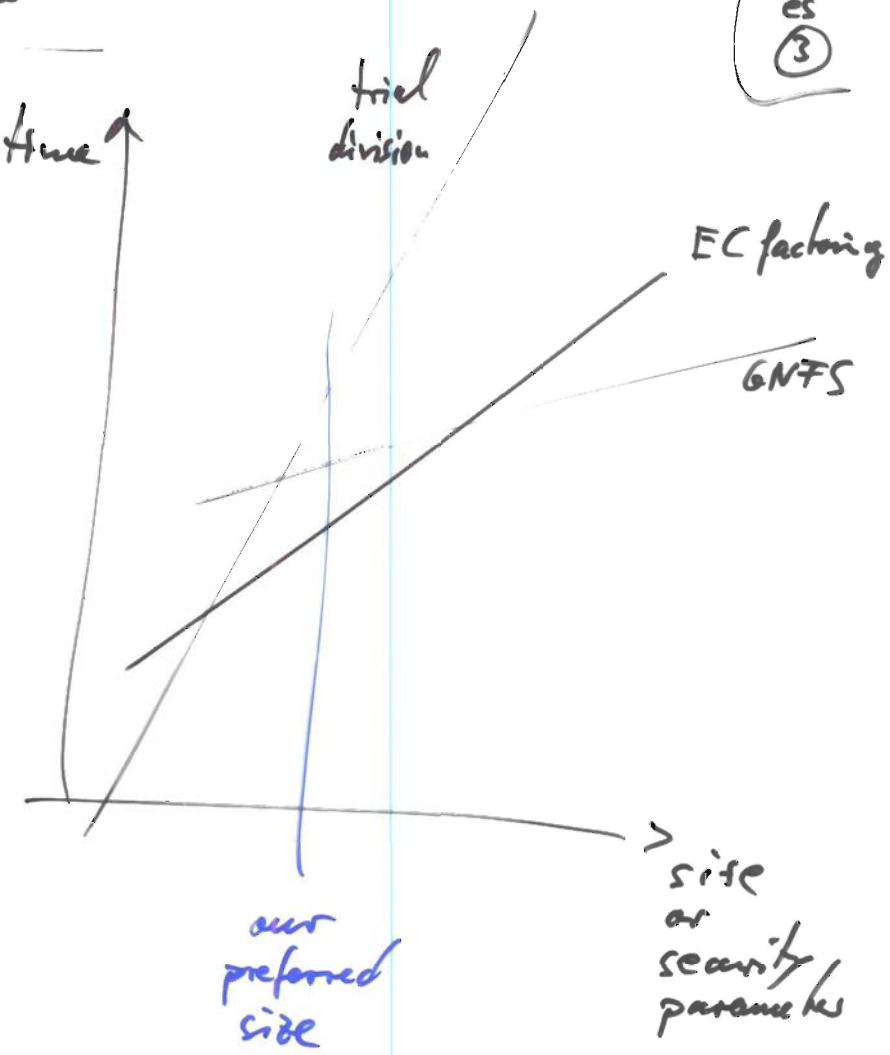
24.4.12
es
(3)

Award an "efficient".

- asymptotic view

versus

- fixed size view



Factoring record:

768 bit, GNFS → 2 yrs, multi parallel.

Award an hybrids

Ex hybrid encryption. RSA 2048 bit → security ≈ 128 bit
AES 128 bit → security ≈ 128 bit.

(Simple) combination has only 64 bit security.

A word on "bits"

→ hard-core bit

in the example of RSA.

Situation: the attacker knows $\underbrace{N, e,}_{\text{public key}}$ $\underbrace{y}_{\text{cipher text}}$.

and shall compute

$$BIT_0(x).$$

[Precisely, we ask for BIT_0 of the least non-negative number representing x.]

Is such an attacker dangerous?

Doesn't seem so ...

BUT it is!

Namely we have the following ~~theorem~~

Theorem

If $\mathcal{B}$ is a poly-time machine that computes $(N, e, y) \longmapsto BIT_0(x)$ with a non-negligible advantage, then there is a poly-time machine $\mathcal{A}$ that computes $(N, e, y) \longmapsto x$ with a non-negligible advantage.

First note that it's easy to achieve a 50% success rate:

GUESSER
Input: N, e, y
Output: $b \in \{0,1\}$.
1. $b \in_R \{0,1\}$
2. Return b.

We assume that $\mathcal{B}$ does better than this!

For simplicity, first assume $\mathcal{B}$ always finds the correct answer.

Let's see: how can we obtain BIT_1, say.

Case $BIT_0(x) = 0$

Then $y = x^e$ in $\mathbb{Z}_N$

and $x = 2x'$.

Of course, $BIT_1(x) = BIT_0(x')$.

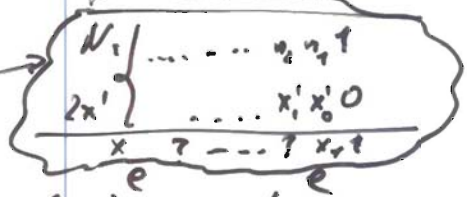
Idea: $y = (2x')^e = 2^e \cdot \underbrace{x'^e}_{=:y'}$

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

is the ciphertext corresponding to x'.

Calling $\mathcal{B}$ gives $BIT_0(x') = BIT_1(x)$.

Case $BIT_0(x) = 1$



Write $x = N - 2x'$,

thus $y = (-2x')^e = (-2)^e \cdot \underbrace{x'^e}_{=:y'}$

and note $BIT_1(x) = BIT_1(N) \oplus BIT_0(x') \quad [\mathrm{mod}\ 2]$.

So calling $\mathcal{B}$ with $(N, e, y \cdot (-2)^{-e})$ reveals $BIT_0(x')$ and thus $BIT_1(x)$.

We can iterate this, or better call it recursively instead of B revealing the entire x' instead of just its BIT_0.

Difficulty: in general B is weaker than our assumption.

Solution: first amplify B such that its success probability $\geq 1 - c^{-k}$.

This can be done with only polynomially many calls to B!

Then the success rate of $\mathcal{A}$ will be something like $\underbrace{(1-c^{-k})^k}_{\approx \exp(-\cdots)}$ exponentially close to 1.

Δ

Conclusion: an attacker obtaining just a single bit of the plaintext may in some cases be able to obtain the entire plaintext.

This is why IND is necessary, where the attacker also just needs to find a single bit.

25.9.12
es 3

So lets check our options:

Is RSA IND-CCA$_2$ secure?

Is ElGamal ?

Theorem RSA is not IND-CCA$_2$ secure.

Pf $\mathcal{A}$ just picks $m_0^* = 0$, $m_1^* = 1$
and the $c^* = 0$ iff $b = 0$,
$c^* = 1$ iff $b = 1$.
So $\mathcal{A}$ can answer correct in 100%
of the cases. □

Pf2 $\mathcal{A}$ picks m_0^*, m_1^* randomly such that $m_0^* \neq m_1^*$.
Then $(m_0^*)^e$ is the enc. of m_0^*
and it is known to $\mathcal{A}$
and $(m_1^*)^e$ is the enc. of m_1^*
and it is known to $\mathcal{A}$.
So just compare c^* to these two to
get 100% correct answer. □

Pf2 generalizes to any encryption scheme
where the encryption is deterministic,
ie. to every plain text there is at most one
cipher text:

Theorem No deterministic encryption scheme
can be IND-CCA$_2$ secure,
(not even IND-KOA secure). □

25.4.12
ex (4)

Theorem ElGamal encryption is not IND-CCA secure

Recall

enc: P base point in a group G,
inputs: A public key,
M message
output: $C = (T, D) \in G^2$ ciphertext

1. Pick $t \in_R \mathbb{Z}_{ord(P)}$, and compute $T = t \cdot P$.
2. Compute $D = M + t \cdot A$.
3. Return $C = (T, D)$

dec(T, D) = $D - a \cdot T$.
where $A = a \cdot P$, a the secret key corr. to the public key A.

Note: this is not deterministic!

This scheme has an additional feature:

Lemma ElGamal encryption is homomorphic, i.e.

$$dec(enc\, M_1 + enc\, M_2) = M_1 + M_2$$

Pf

$enc\, M_1 = (T_1, D_1)$
where $T_1 = t_1 \cdot P$, $D_1 = M_1 + t_1 A$
$enc\, M_2 = (t_2 P, M_2 + t_2 A)$

thus $enc\, M_1 + enc\, M_2 = ((t_1+t_2)P, M_1 + M_2 + (t_1+t_2)A)$

and

$$dec(enc\, M_1 + enc\, M_2) = [M_1 + M_2 + (t_1+t_2)(aP)] - a \cdot [(t_1+t_2)P]$$
$$= M_1 + M_2$$

□

~~Note~~ One can use this to reencrypt message by applying the lemma with $M_2 = 0$.

ES.4.12
es
④

Pf (ElGamal is not IND-CCA secure).

$\mathcal{A}$ picks m_0^*, m_1^* at random, different.
Then he ask for an encryption c^* of one of them.
And now he recrypts c^*: $c_2 = c^* + \text{enc } 0$.
Pass this to decryption oracle and obtain
its decryption m_2. But this is the same
plaintext as the one corr. to c^*. Thus

if $m_2 = m_0^*$ answer 0,

if $m_2 = m_1^*$ answer 1.

else: haven't the challenge oracle
(can never happen!!

□

this attacker again is efficient and has 100% success. □

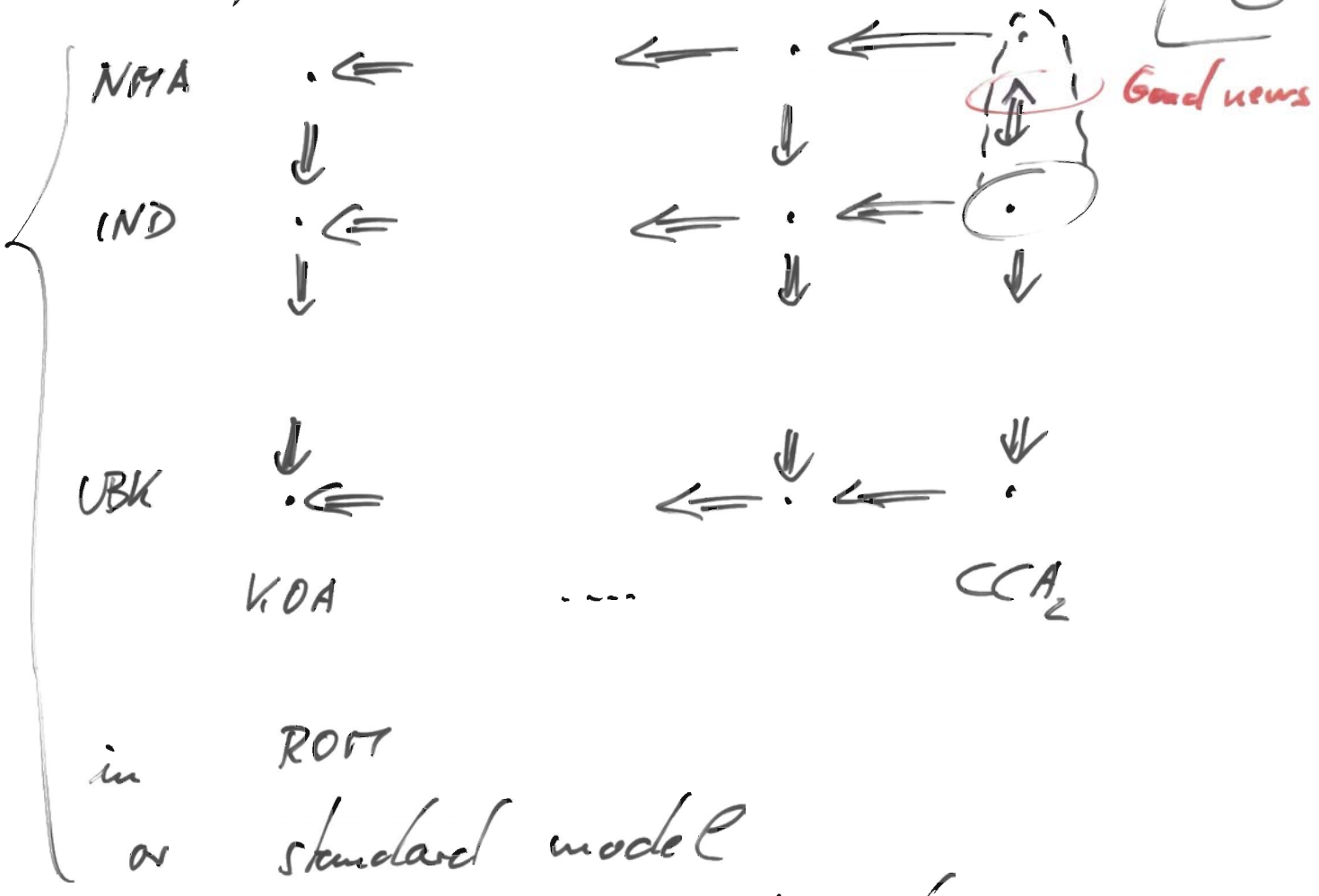
Thm ElGamal encryption actually
is IND-~~KOA~~ CPA secure,
provided the ~~DLP~~ DDH of the used group
is difficult. ∎

To save the world: use a non-structured, reversible
operation when encoding M!
For example, use

$$D = M \oplus h(tA)$$

where h is suitable hash function.
... This yields (I think) an IND-CCA$_1$ secure
encryption scheme in ROM.

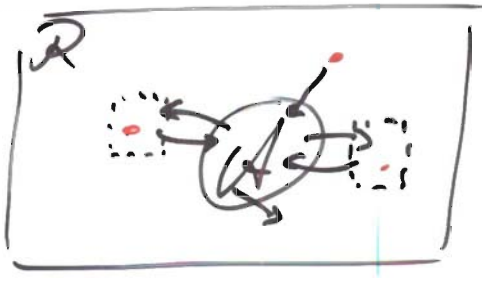
Summary



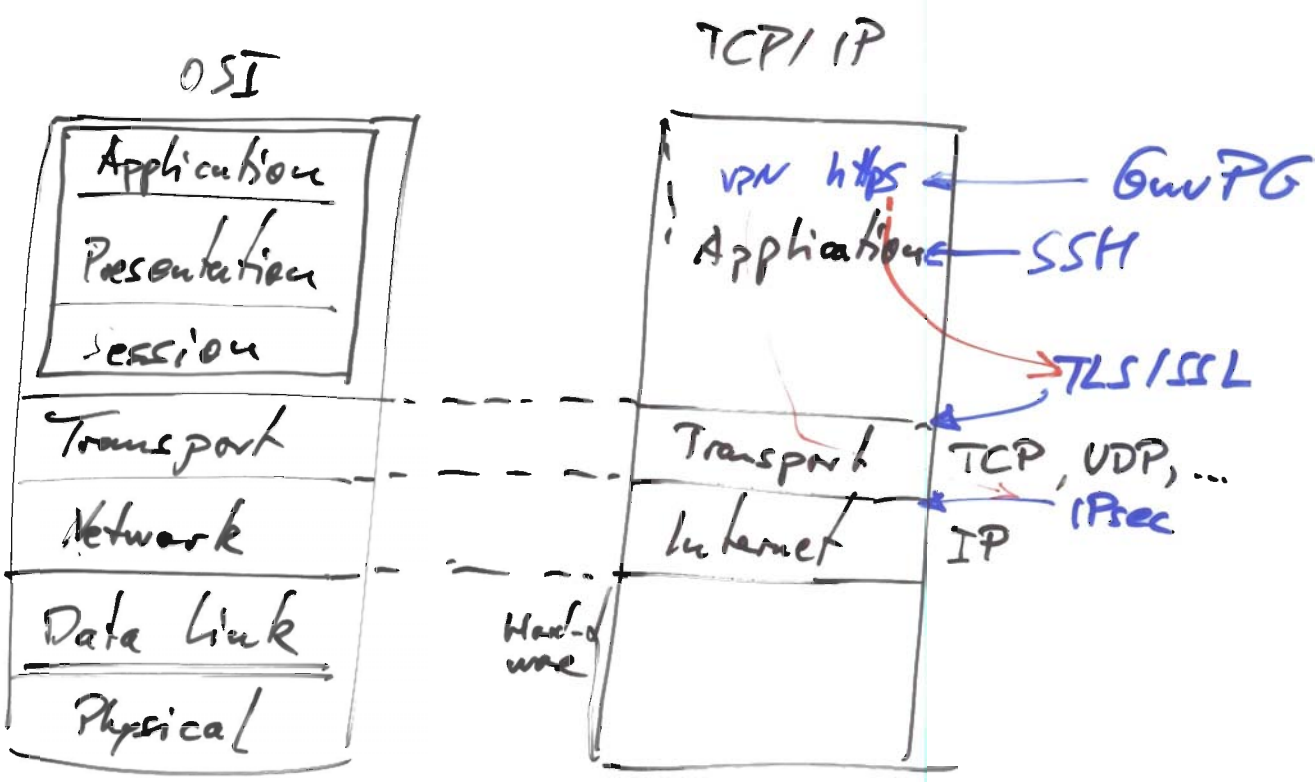
Security notions for encryption schemes.

There is a similar picture for security notions of signature schemes.

Typical security ~~proofs~~ reductions do this:
Assume $\mathcal{A}$ is a successful attacker.
Construct a protocol $\mathcal{R}$ (reduction)
that may use $\mathcal{A}$ and solves
some problem P



Secure connections



Q: What about wireless security protocols? Where to put them?

Pro/cons of the placement in the layer model?

- The lower/deeper the security the more difficult it becomes for a firewall to check.
- The lower — the more is protected.
- The lower — the faster !? the slower?
- The lower — the more software can use it.
- The higher — the more control inside the application does
- Implementation: the lower the more upper layer stuff may have to be redone.
  the easier / more difficult...

8.6.12
es
(1)

IPsec

we have combination of three protocols:

IKE = Internet Key Exchange

old IKEv1

new IKEv2

comprises

(1) the exchange of a seed key,

(2) the authentication of the communicating parties,

(3) the choice of algorithms and further parameters, ...

IPsec → AH authentication header

→ allows to authenticate data and some parts of the IP header.

→ ESP encapsulated security payload

→ allows encryption

→ allows authentication (w/o header)

8.5.12
es
(2)

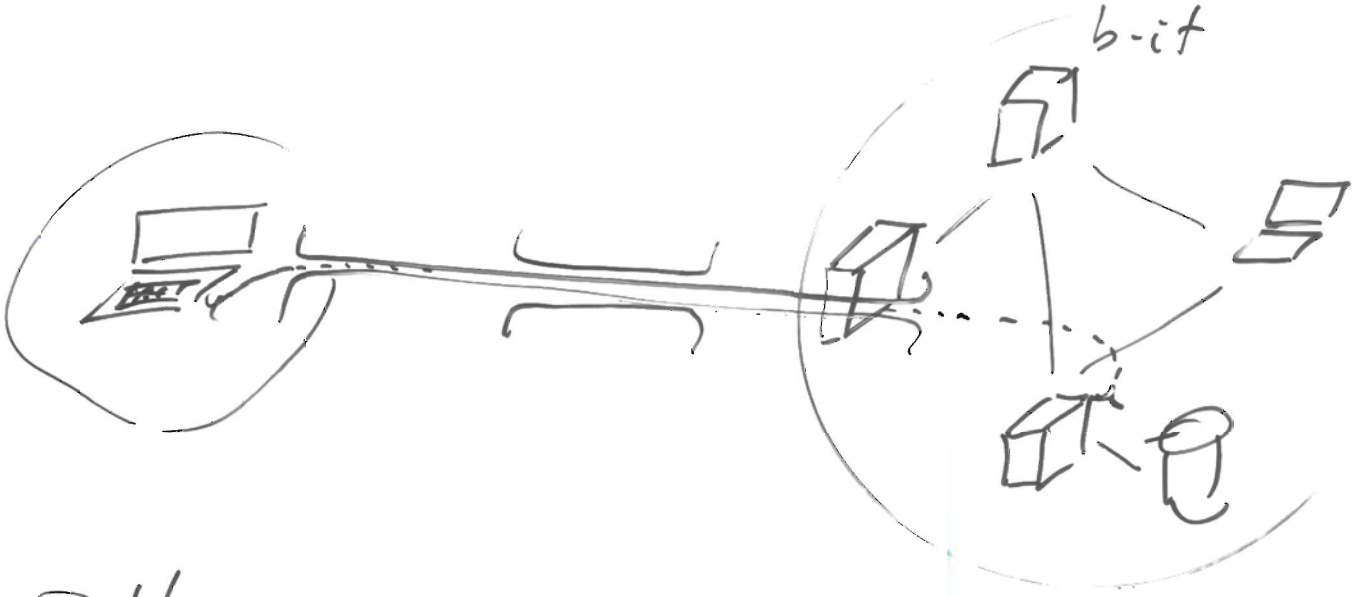
Scenarios for tunnel

Transport modes:

- tunnel mode
- transport mode

In tunnel mode a new IP header can be used with different info (in particular source & destination) than the original IP header.

So we may realize scenarios like this:



Problems

IPsec ESP in tunnel mode

- No stateful inspection of higher layer info.
- NAT is problematic.

This is why we have AH and transport mode.

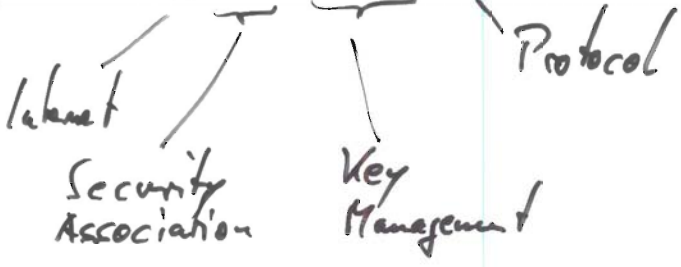
↳ Many, many options. (IPv6.)

TSicherheit 6: IPv4 in Ideo. of IPsec but IPsec is wanted

History of IKE

PHOTURIS ↯ SKIP

NSA proposed ISAKMP



- not a protocol, just a framework.

Con:
- Development took much too long.
- No clear design.
- Too many variants.

≥ 3 RFC

difficult to read, notation changes even

IPSEC & IKE

Michael Nüsken

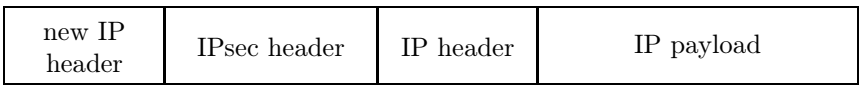
25 June 2007

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.

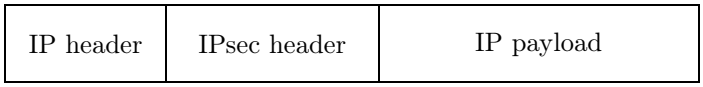
1. IPsec

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

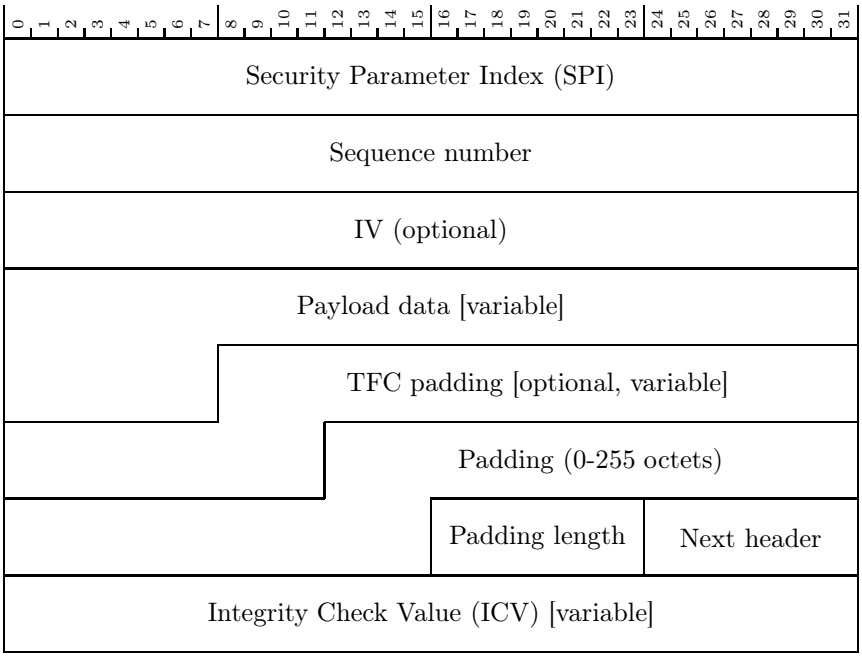


In transport mode, only the IPsec header is added:



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:

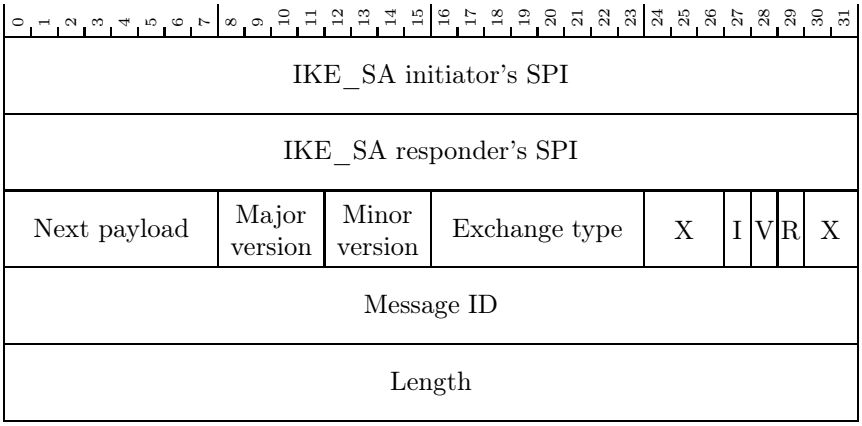


The security parameter index identifies the SA and thus all necessary algorithms and key material. To create the secured packet from the original one, it is first padded. Padding is used to enlarge the data length to a multiple of a block size that might be associated with the encryption. Traffic flow 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.)

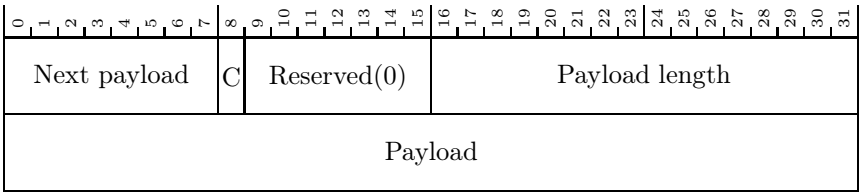
2. Internet key exchange (version 2)

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



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

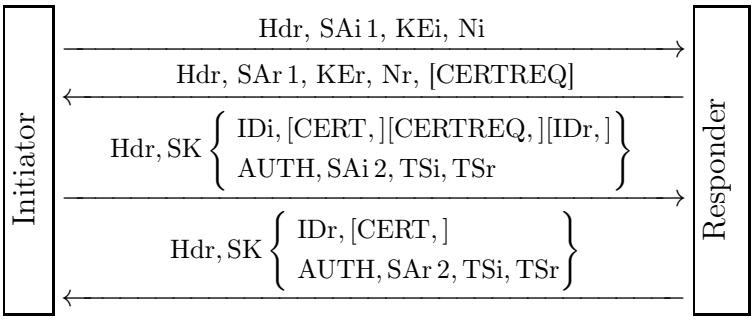
Exchange type	Value
Reserved	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
Reserved to IANA	38-239
Reserved for private use	240-255



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:

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

.)
Choose a nonce Nr.

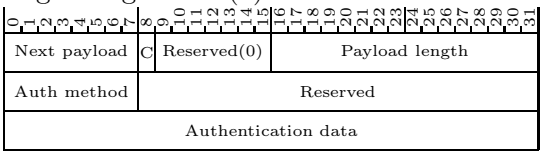
$$\xleftarrow{\text{Hdr, SAr 1, KEr, Nr, [CERTREQ]}}$$

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.

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

where $\text{prf+}(K, S) = T_1|T_2|T_3|\ldots$, and $T_1 = \text{prf}(K, S|0x01)$, $T_i = \text{prf}(K, T_{i-1}|S|i)$ 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.

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

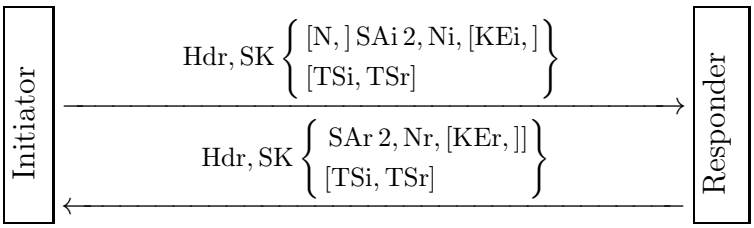
5. The responder sends its identity IDr, certificate(s). He computes an authentication AUTH for the entire second message concatenated with the initiator's nonce Ni and the value prf(SK_pr, IDr). Further he supplies the answer SAr 2 to the child SA creation and sends the accepted traffic selectors TSi, 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.

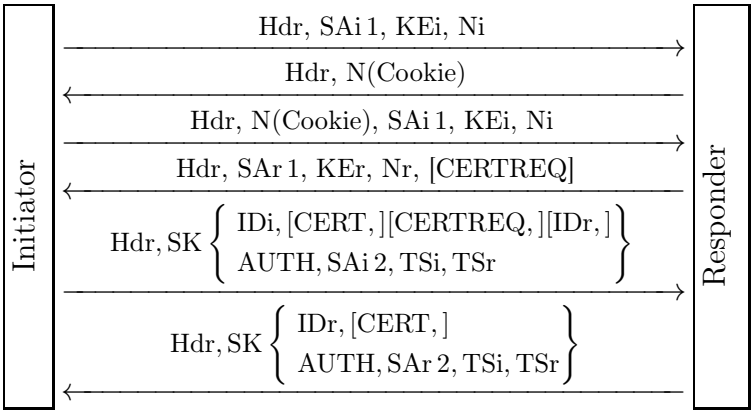
In a CREATE_CHILD_SA exchange including an optional Diffie-Hellman exchange new keying material uses also the new Diffie-Hellman key g^{ir}, it is concatenated left to the nonces. (Though the Diffie-Hellman key exchange is optional, it is recommended to either used it or at least to limit the number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

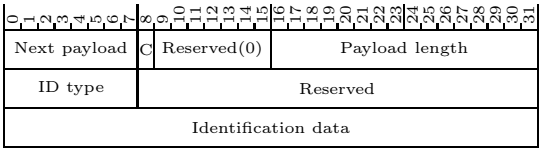
The authentication AUTH then refers to the second version of the corresponding message, so the one including the cookie or responding to that, respectively. So the protocol becomes:



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.

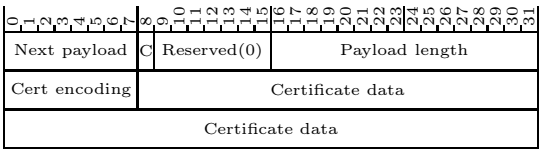
2.5. IP compression. The parties can negotiate IP compression.

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.

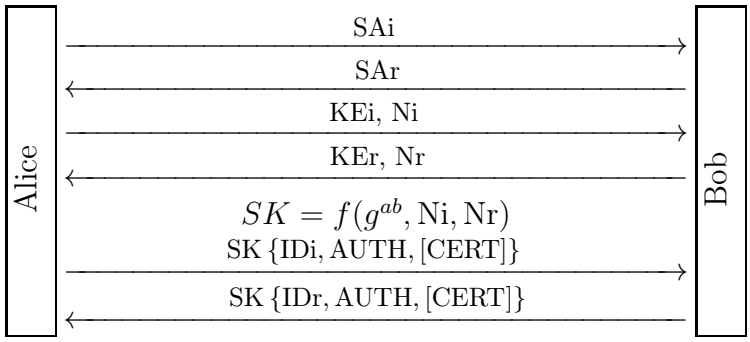
3. IKE version 1

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

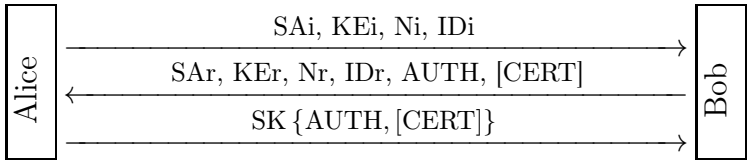
- public signature keys,
- public encryption keys, originial protocol,
- public encryption keys, revised protocol, or
- a pre-shared secret.

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 Kaufmann *et al.* 2002.)

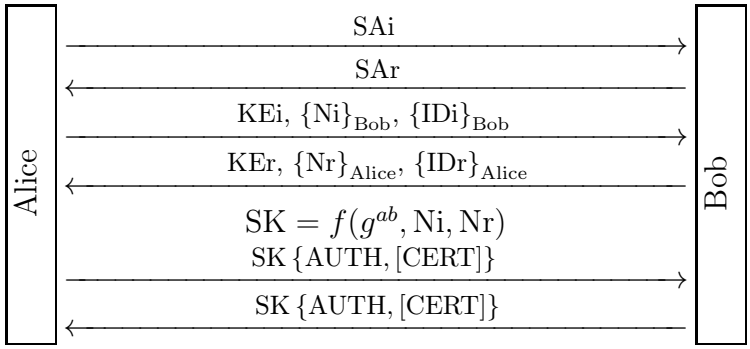
3.1. Main mode, public signature keys.



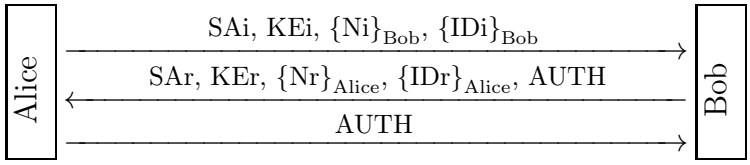
3.2. Aggressive mode, public signature keys.



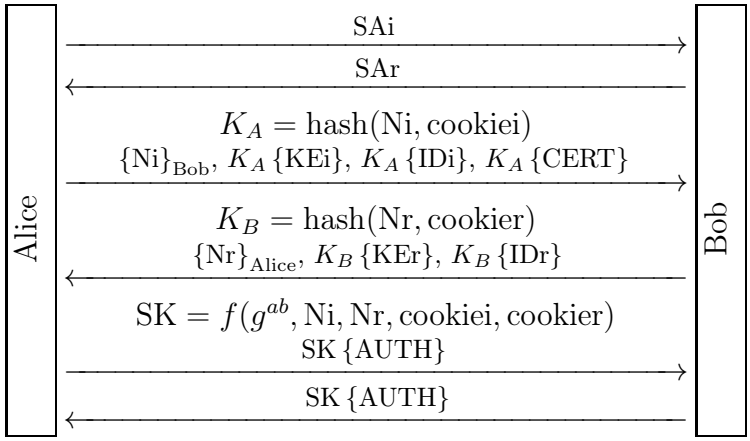
3.3. Main mode, public encryption keys, original protocol.



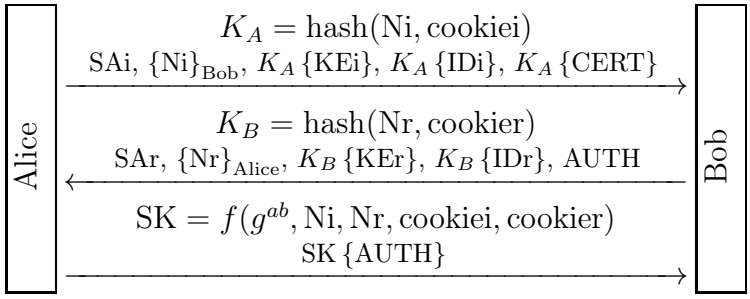
3.4. Aggressive mode, public encryption keys, original protocol.



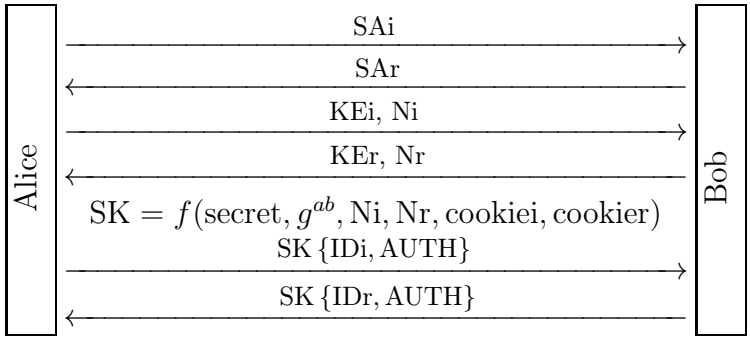
3.5. Main mode, public encryption keys, revised protocol.



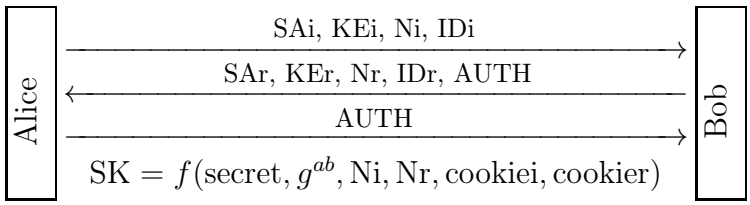
3.6. Aggressive mode, public encryption keys, original protocol.



3.7. Main mode, pre-shared secret.



3.8. Aggressive mode, pre-shared secret.



References

Charlie Kaufmann, Radia Perlman & Mike Speciner (2002). *Network Security.* Prentice-Hall, Inc., New Jersey. ISBN 0-13-046019-2.

Michael Nüsken
b-it, Bonn, Germany

Security questions

9.5.12
(4)
es

(6) Secure?
By what definition? → Defer.

(1) Session key agreement.

- How long? Random? Unpredictable?

> If the seed used for key generation is too short then the scheme is easily broken.
>
> (↑ Debian openssl fiasco).

For IPsec it depends on the Diffie-Hellman group and the key generation based on it.

- DH group ≥ 160 bit (for 80 bit security)
- prf output ≥ 160 bit (for 80-bit security)

Thus we can assume that there is enough entropy (provided of course, you use enough entropy).

- Do both parties contribute to it?
  IPsec: Yes!
  Note: this is important to ensure security for each party.

• Man in the middle?

IPsec: Not possible due to authentication

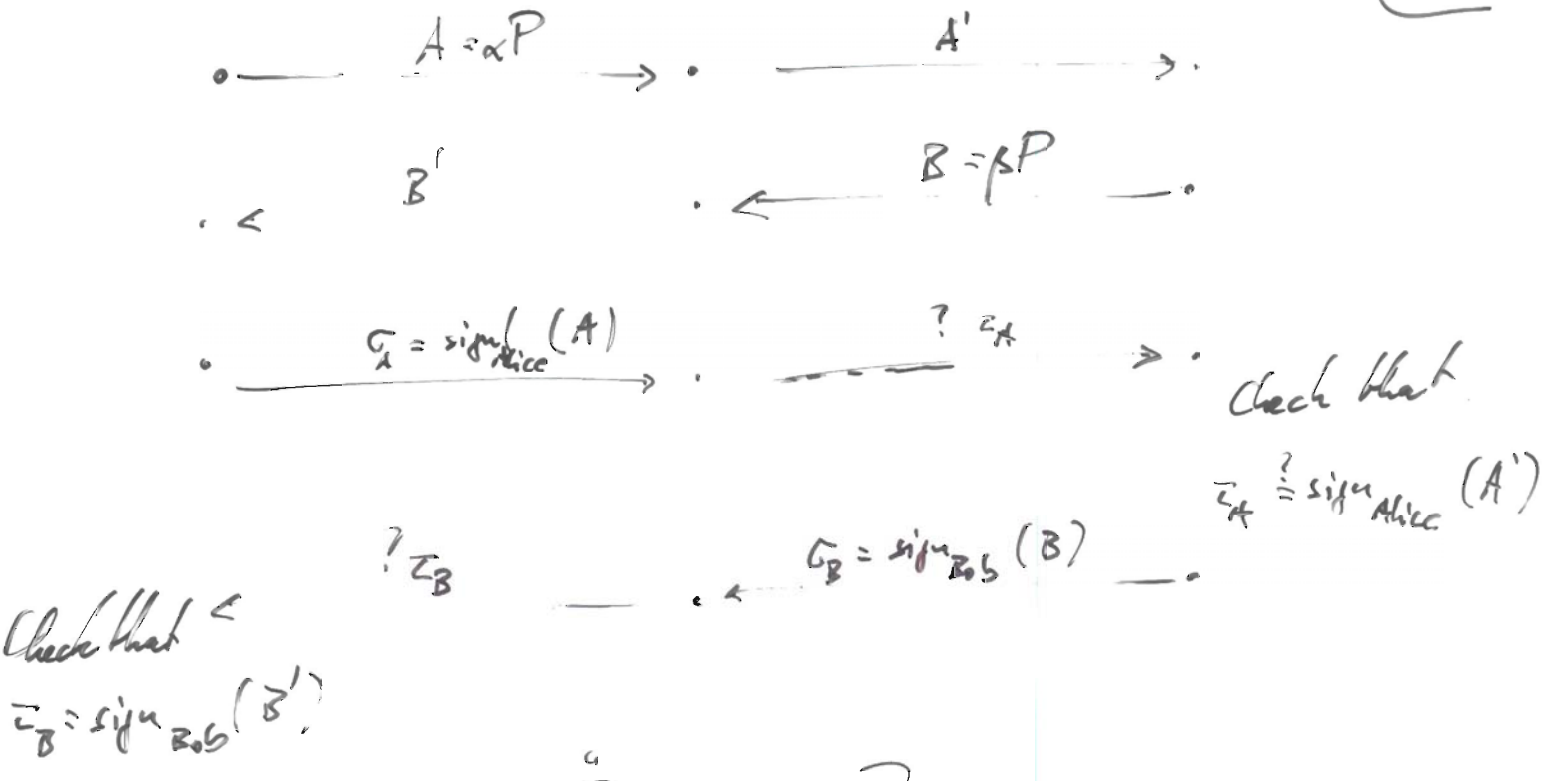
unless ...

(9.5.12 es (2))

Of course, authentication always requires some knowledge about the other party. This can be provided by certificates secured in a PKI, trusted by both parties.
(→ Security anchors)

Basic idea:

(15.5.12 es (1))



? αβP ?

Alice can break if he can solve the DH-problem ... break the signature scheme

(2) Perfect forward security „Beagle boys"

15.5.12
es. (2)

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

IPsec: No, the attacker only finds the secret keys used for the AUTHentication but no remains of the short-term session keys which were entirely generated based on IKE.

GnuPG: Yes, the attacker can decrypt (provided he can bypass login password and passphrase)

So here: no Forward Security.

To the contrary: IPsec does provide Perfect Forward Security.

Escrow failage

- Can an attacker decrypt ongoing conversations if she is in possession of the long-term secrets?

GnuPG: Yes. No escrow failage here.

IPsec: If the attacker was active as a man in the middle then: ~~Yes~~... Partially ... no. Unless the attacker is escrow to both parties.

Passive case: No, we have Escrow failage.

15.5.12
es
(3)

(3) Denial of service

→ DoS, DDoS, DDoS via botnet, ...

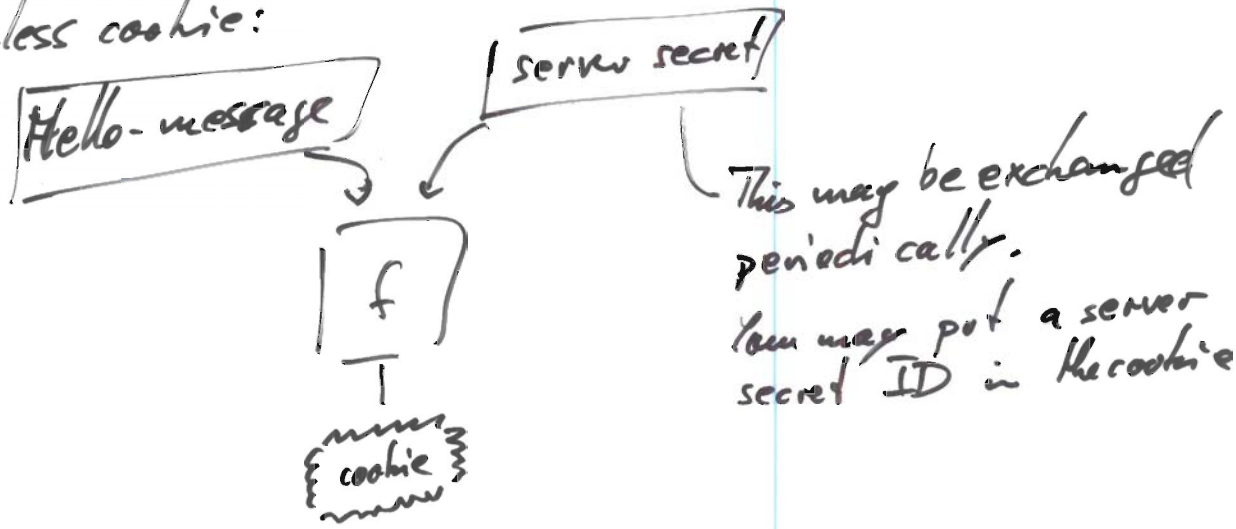
- We cannot prevent this completely, but we can prevent memory overflow due to many half-open connections by the cookie construction in IPsec.

First defense: increase table sizes.
Second defense: cookies.
Third defense: stateless cookies (as in IPsec)

Basic feature of cookies: they are unpredictable!

Otherwise the attacker could predict the cookie and directly send it and thus bypass the defense.

Stateless cookie:



Pros: The server only needs a very fast computation and no storage.

Important: f must be deterministic suitable unpredictable pseudo-random

- server secret is unpredictable.

Typically: f is hash function...

(4) Endpoint identifier hiding

- Does an eavesdropper get information about the identities?

  In IPsec the ID and CERT are usually encrypted and thus not visible to an attacker, unless she can break the encryption.

- Can an active attacker collect identity information?

  in principle, in one direction you cannot hide the identity. At best, you can hide one identity, initiator or receiver. But never both. One party has to reveal its identity first.

  IPsec: hides the responder ID.

  The responder gets the initiator ID and CERT before he needs to reveal his own and thus can decide upon this information.

22.5.2012
ES
(1)

(5) Live partner reassurance

- Is a replay attack possible?

Since nonces are used in the key generation it is intractable to force the same key material. So we can assume live partners if we make nonce a new number every time. And after that no replay will work.

(6) Plausible deniability.

- Does the protocol of a session prove
  - that Alice talked? No.
  - ~~that~~ to Bob that Alice talked? Yes.
  - that (Alice or Bob) talked? Yes.

in IPsec provided we use authentication in ESP or AH.

(7) Stream protection.

- How is the stream protected?

	IPsec
→ Confidentiality:	Yes, unless no encryption is used.
→ Integrity	Yes, unless no authentication is used
→ Authenticity	

(8) Negotiating parameters?

IPsec: Yes!

Pro:
- flexible
  We exchange a cryptographic primitive with a better one. In particular, if one is broken.

  Eg. DES can be brute-force broken within a day by trying all 2^{56} keys.

Con:
- complexity
  IPsec has a clear, simple structure and thus still be analyzed easy enough.
- „downgrading" by attackers
  IPsec counters this by repeating the partner again and again if you could use better protocol versions or algorithms.
- admins may choose bad by :-

Information about algorithms:

Kerckhoffs' principle

The only unknown thing to the attacker is the (secret) key.

3. Algorithm Selection

3.1. IKEv2 Algorithm Selection

3.1.1. Encrypted Payload Algorithms

The IKEv2 Encrypted Payload requires both a confidentiality algorithm and an integrity algorithm. For confidentiality, implementations MUST- implement 3DES-CBC and SHOULD+ implement AES-128-CBC. For integrity, HMAC-SHA1 MUST be implemented.

3.1.2. Diffie-Hellman Groups

There are several Modular Exponential (MODP) groups that are defined for use in IKEv2. They are defined in both the [IKEv2] base document and in the MODP extensions document. They are identified by group number. Any groups not listed here are considered as "MAY be implemented".

Group Number	Bit Length	Status	Defined
2	1024 MODP Group	MUST-	[RFC2409]
14	2048 MODP Group	SHOULD+	[RFC3526]

3.1.3. IKEv2 Transform Type 1 Algorithms

IKEv2 defines several possible algorithms for Transfer Type 1 (encryption). These are defined below with their implementation status.

Name	Number	Defined In	Status
RESERVED	0		
ENCR_3DES	3	[RFC2451]	MUST-
ENCR_NULL	11	[RFC2410]	MAY
ENCR_AES_CBC	12	[AES-CBC]	SHOULD+
ENCR_AES_CTR	13	[AES-CTR]	SHOULD

3.1.4. IKEv2 Transform Type 2 Algorithms

Transfer Type 2 Algorithms are pseudo-random functions used to generate random values when needed.

Name	Number	Defined In	Status
RESERVED	0		
PRF_HMAC_MD5	1	[RFC2104]	MAY
PRF_HMAC_SHA1	2	[RFC2104]	MUST
PRF_AES128_CBC	4	[AESPRF]	SHOULD+

3.1.5. IKEv2 Transform Type 3 Algorithms

Transfer Type 3 Algorithms are Integrity algorithms used to protect data against tampering.

Name	Number	Defined In	Status
NONE	0		
AUTH_HMAC_MD5_96	1	[RFC2403]	MAY
AUTH_HMAC_SHA1_96	2	[RFC2404]	MUST
AUTH_AES_XCBC_96	5	[AES-MAC]	SHOULD+

4. Security Considerations

The security of cryptographic-based systems depends on both the strength of the cryptographic algorithms chosen and the strength of the keys used with those algorithms. The security also depends on the engineering of the protocol used by the system to ensure that there are no non-cryptographic ways to bypass the security of the overall system.

This document concerns itself with the selection of cryptographic algorithms for the use of IKEv2, specifically with the selection of "mandatory-to-implement" algorithms. The algorithms identified in this document as "MUST implement" or "SHOULD implement" are not known to be broken at the current time, and cryptographic research so far leads us to believe that they will likely remain secure into the foreseeable future. However, this isn't necessarily forever. We would therefore expect that new revisions of this document will be issued from time to time that reflect the current best practice in this area.

5. Normative References

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[IKEv2] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC2451] Pereira, R. and R. Adams, "The ESP CBC-Mode Cipher Algorithms", RFC 2451, November 1998.

2.1. Suite "VPN-A"

This suite matches the commonly used corporate VPN security used in IKEv1 at the time of this document's publication.

IPsec:	
Protocol	Encapsulating Security Payload (ESP) [RFC2406]
ESP encryption	TripleDES in CBC mode [RFC2451]
ESP integrity	HMAC-SHA1-96 [RFC2404]

IKE and IKEv2:	
Encryption	TripleDES in CBC mode [RFC2451]
Pseudo-random function	HMAC-SHA1 [RFC2104]
Integrity	HMAC-SHA1-96 [RFC2404]
Diffie-Hellman group	1024-bit Modular Exponential (MODP) [RFC2409]

Rekeying of Phase 2 (for IKE) or the CREATE_CHILD_SA (for IKEv2) MUST be supported by both parties in this suite. The initiator of this exchange MAY include a new Diffie-Hellman key; if it is included, it MUST be of type 1024-bit MODP. If the initiator of the exchange includes a Diffie-Hellman key, the responder MUST include a Diffie-Hellman key, and it MUST of type 1024-bit MODP.

2.2. Suite "VPN-B"

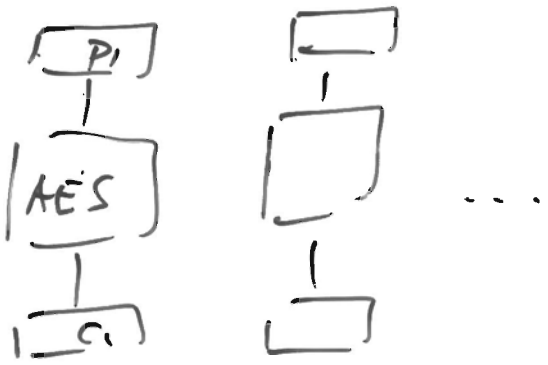
This suite is what many people expect the commonly used corporate VPN security that will be used within a few years of the time this document's publication.

IPsec:	
Protocol	ESP [RFC2406]
ESP encryption	AES with 128-bit keys in CBC mode [AES-CBC]
ESP integrity	AES-XCBC-MAC-96 [AES-XCBC-MAC]

IKE and IKEv2:	
Encryption	AES with 128-bit keys in CBC mode [AES-CBC]
Pseudo-random function	AES-XCBC-PRF-128 [AES-XCBC-PRF-128]
Integrity	AES-XCBC-MAC-96 [AES-XCBC-MAC]
Diffie-Hellman group	2048-bit MODP [RFC3526]

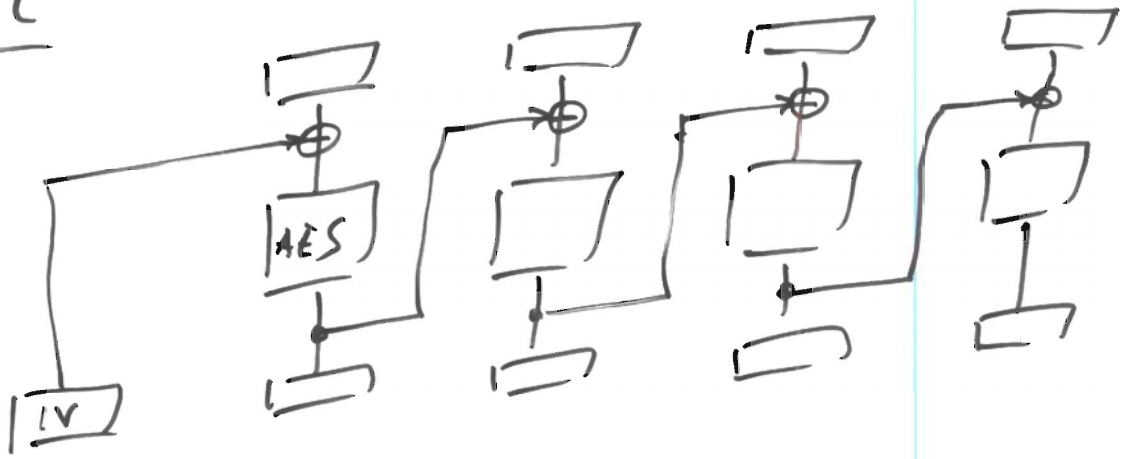
Rekeying of Phase 2 (for IKE) or the CREATE_CHILD_SA (for IKEv2) MUST be supported by both parties in this suite. The initiator of this exchange MAY include a new Diffie-Hellman key; if it is included, it MUST be of type 2048-bit MODP. If the initiator of the exchange includes a Diffie-Hellman key, the responder MUST include a Diffie-Hellman key, and it MUST of type 2048-bit MODP.

ECB



Con: Same blocks are encrypted with the same cipher text all the time.

CBC



Pro: · Blocks are encrypted all the time
· IV allows many encryptions of the same plain text

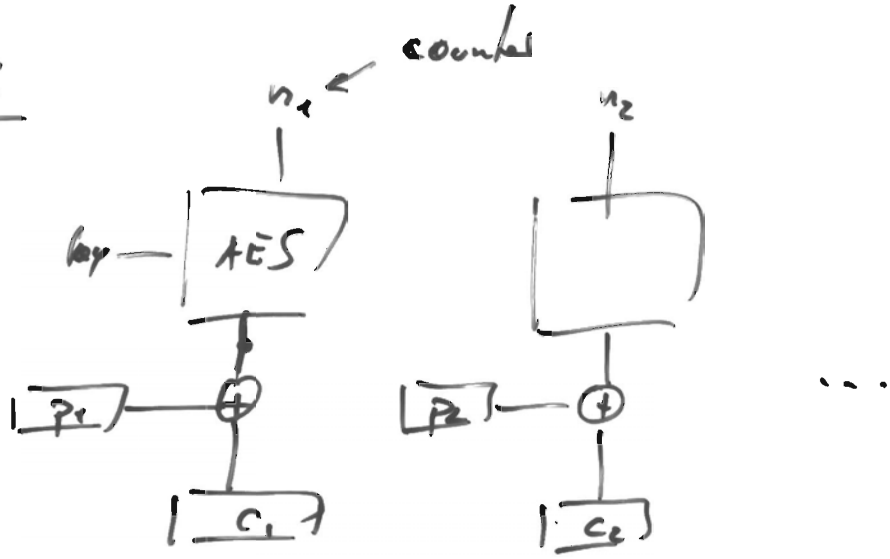
Pro: If one ciphertext is lost, it resynchronizes very fast, only one other block is garbled.

Con: We must receive stuff in order. Otherwise we have buffer those faster blocks.

Pro: There is a security ~~Proof~~ reduction.

22.5.12
es

CTR



Pro : • There is a security reduction.
- It resynchronizes immediately.
- We can precompute this  value
- Blocks may be dealt in any order.

HMAC

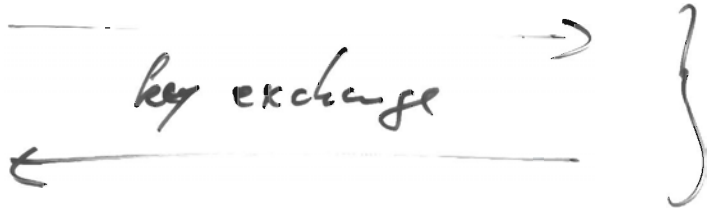
Pro : • There is a security reduction.

AES-CBC

Pro : • There is a security reduction
- May ~~use~~ reuse an encryption primitive.

A secure connection

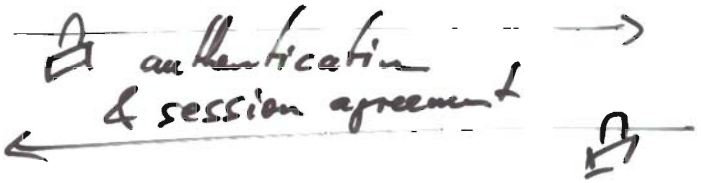
27.5.12
es
(5)



Diffie-Hellmann

need:

- „secure" group
- good (pseudo) random number generator (& source of truly random bits) ↗ unpredictable to an attacker.



need:

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

need:

- FAST algorithms:
  - fast encryption ⇒ symm.
    eg. AES
  - fast authentication + integrity ⇒ symm.
    eg. HMAC-SHA1
    AES-CBC-MAC

Back to HMAC-SHA1

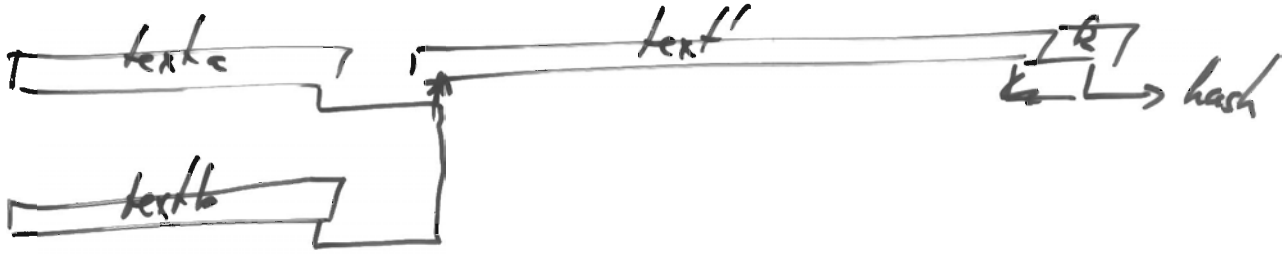
① Can we omit the key at the end?

No: EXTENSION attack!

extended message

② Can we omit the key at the beginning?

Assume that you have a collision for the hash function. Then



So a collision for the unkeyed hash function will also be a collision for this 'MAC'.

Essentially, this points to the right

Security for a keyed hash function (or message authentication code)

(eff.) No attacker shall be able to find a collision – without knowing the key.

This is very similar to collision resistence of (unkeyed) hash functions.

We shouldn't simplify the attacker's task by omitting the key at the beginning.

5.6.12
es
(2)

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

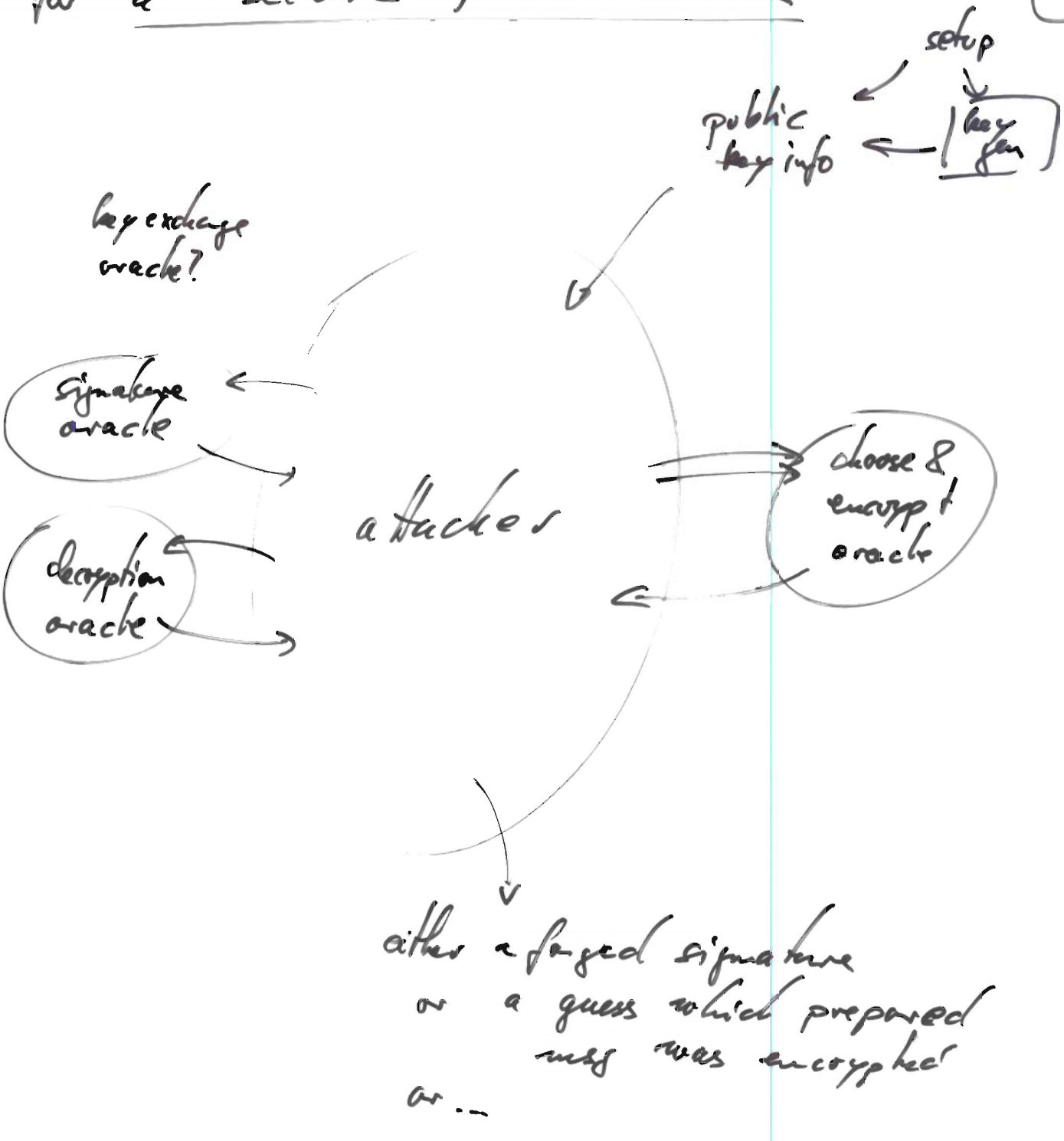
Let's consider the extreme: half of the key is used at the beginning and (the other half) at the end.

Say: 160 bit key.
We expect security 2^{160}.
by brute-force.
By „splitting in the middle" this can now be reduced to $2 \cdot 2^{80}$.
At least this would work if the compression-function is a block cipher encryption.

„Theorem"

If the used hash function is suitably secure the the HMAC construction based on it is also secure.

Possible security definition for a secure protocol connection



Difficulty: This is a composed security assumption.

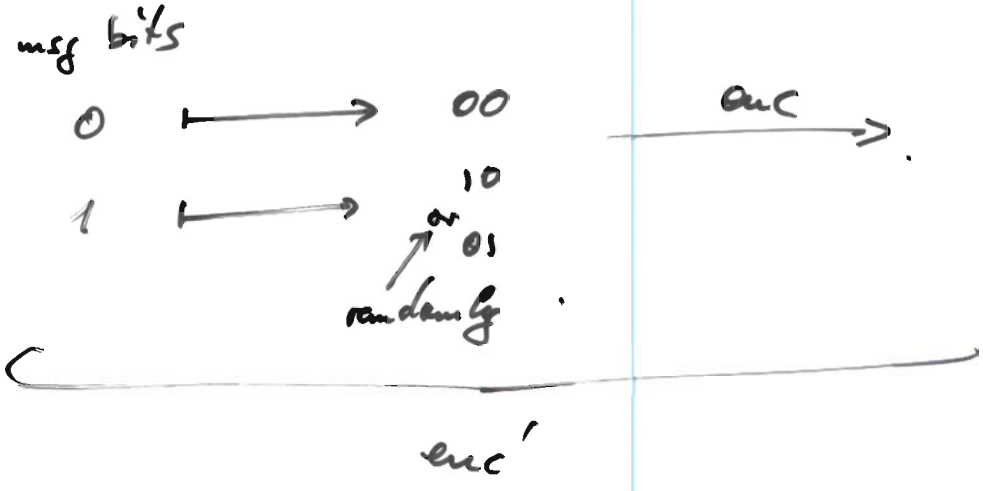
15.6.17
es
(3)

Example

Assume we have an secure (IND-CCA or similar) symmetric encryption scheme enc which works xoring the plaintext with some generated key stream.

Further assume that we use a MAC on the plaintext.

Replace the encryption by enc':



The attacker sends the password message with the ~~bit~~ first two bits inverted.

If the first plaintext bit was 0 then this changes the ~~00~~ intermediate plaintext 00 to 11 which does never occur in a legal transmission. So the server sends an error message.

If the first plaintext bit was 1 then it merely exchanges 01 with 10 which both are encodings of 1 and so the server accepts the password.

E Horton hears a who!

5.6.11
es
(4)

Horton's principle

> A signature or authentication value must depend on the meaning of the plain text.

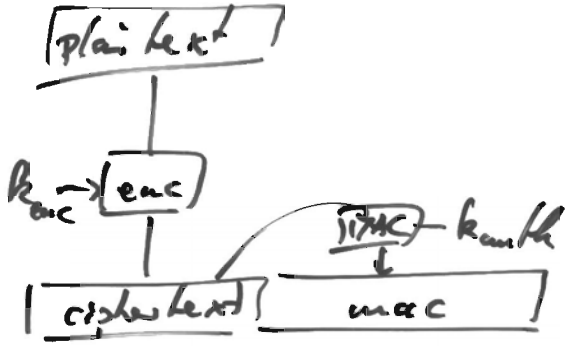
Now: How to combine ~~the~~ encryption and authentication?

Option

- A t E: authenticate, then encrypt.
- E t A: encrypt, then authenticate.

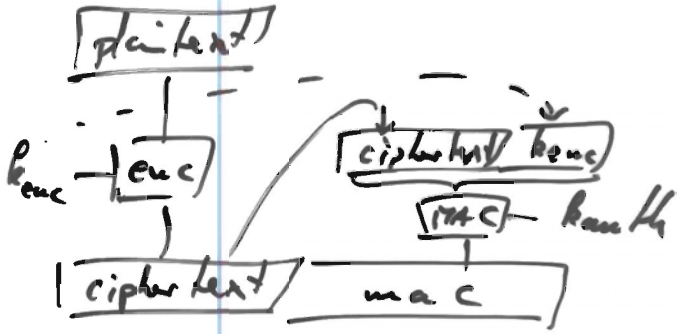
(1) E t A violates Horton's principle.

a)



This violates Horton because the ciphertext belongs to many different plaintexts.

b)



This is ok to Horton because ciphertext + k_{enc} identify the plaintext.

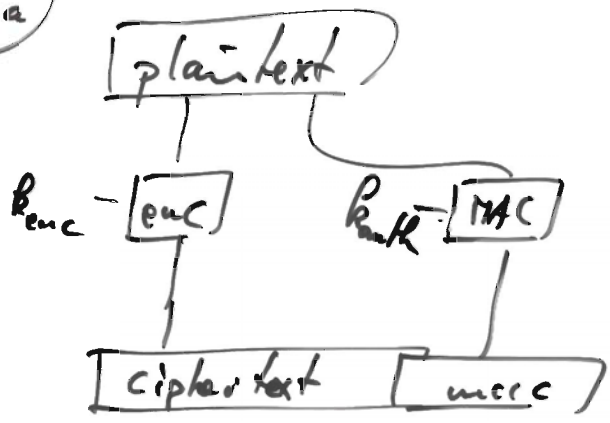
pro: The recipient saves the decryption work if the authentication fails.

But: IPsec uses E&A ②.
Bad news, as this seems to violate Horton's principle.

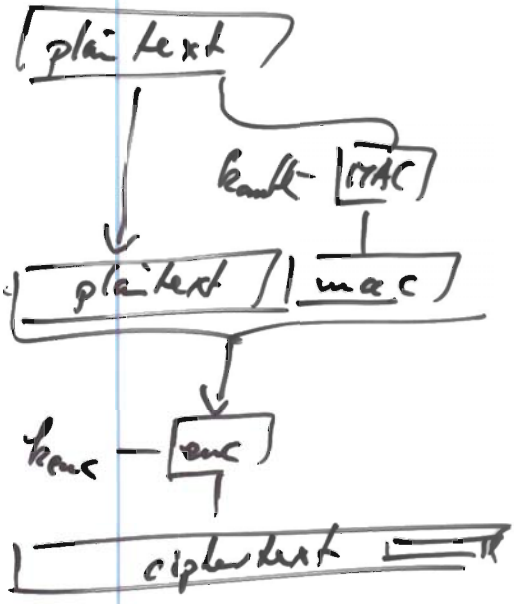
The point is that one cannot change the ~~encryption~~ authentication key without affecting the authentication key or breaking the pseudo random function used in the key generation.

AtE

2a



2b



CON: The recipient must decrypt before he can check the authentication.

TLS/SSL:

SSH:

6.6.17
ec
①

Part II

€

what is money?

coin

trade

value

promise
contract
Law
Punishment

acquire / spend
pay

gold
equivalence
/
economy
figures

bank

hard to replicate

Scottish money reads sth like
„Show this banknote to ...
and you'll get ...“

6.6.12
es
(2)

Forms of money

- coins
- banknotes

} hardware money

- bank account
  checks
  ⋮
- stocks

} software money

On the internet? e-currencies?

→ usually account-based transactions
(withdrawal, bank transfer)

Distinction

- traceable / untraceable
- online / offline
- instantaneous / with delay
- anonymous / non-anonymous
- transferable / intransferable

? anonymous + electronic ?

difficult to replicate?

6.6.17
es
(3)

Do we want anonymity?

- Yes, to prevent the account holder (bank) to gather too much information about its customers.
- Yes, to protect privacy.
- No, to be able to trace money related to crimes. in other words: to protect public/social security.
- No, because of the possibility of loss.

Schneier about privacy vs. security

" Freedom is fragile.
Handle it with care. "

Physical cash identifiers

- special printing
  - uv-~~print~~ ink
  - watermark
  - structured paper/relief print
  - serial number
  - silver strip
  - pictures made of (drawn with very thin lines, many of them
  - front and back printing fit perfectly together
  - characteristic colour
- amount
- currency
- bank name
- signature of a representative

6.6.12
ex
(4)

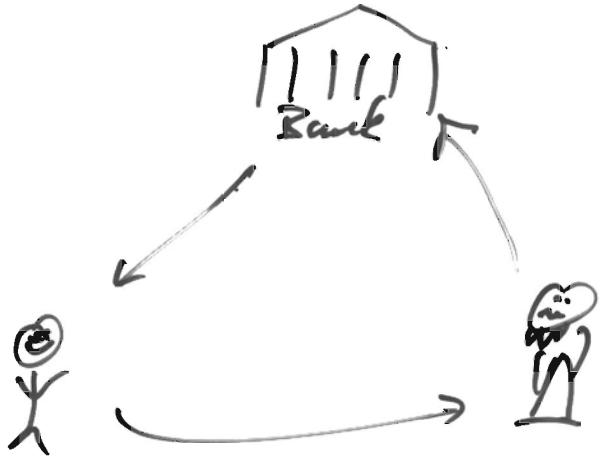
So now we know what essential features are.

We want:

e-cash

- anonymous
- hard to copy: double spending protection
- offline

basic scenario:



The big problem is:

- bitstrings are
  - (1) easy to copy
  - (2) easy to compare / recognise

... blind signatures ...

TLS-HANDSHAKEPROTOCOL

Client

SERVER

CLIENT HELLO { Protocol version; Supported chipers (List); Supported compression (List); Client random 28 ~~Bytes~~ } Preferred order →

← SERVER HELLO { PROTOCOL VERSION; SELECTED CIPHER SUITE; SELECTED COMPRESSION ALGORITAM; RANDOM DATA (28B) }

← CERTIFICATE ?
← SERVER KEY EXCHANGE ?
← CERTIFICATE ~~VERIFY~~ REQUEST ?
← SERVER HELLO DONE

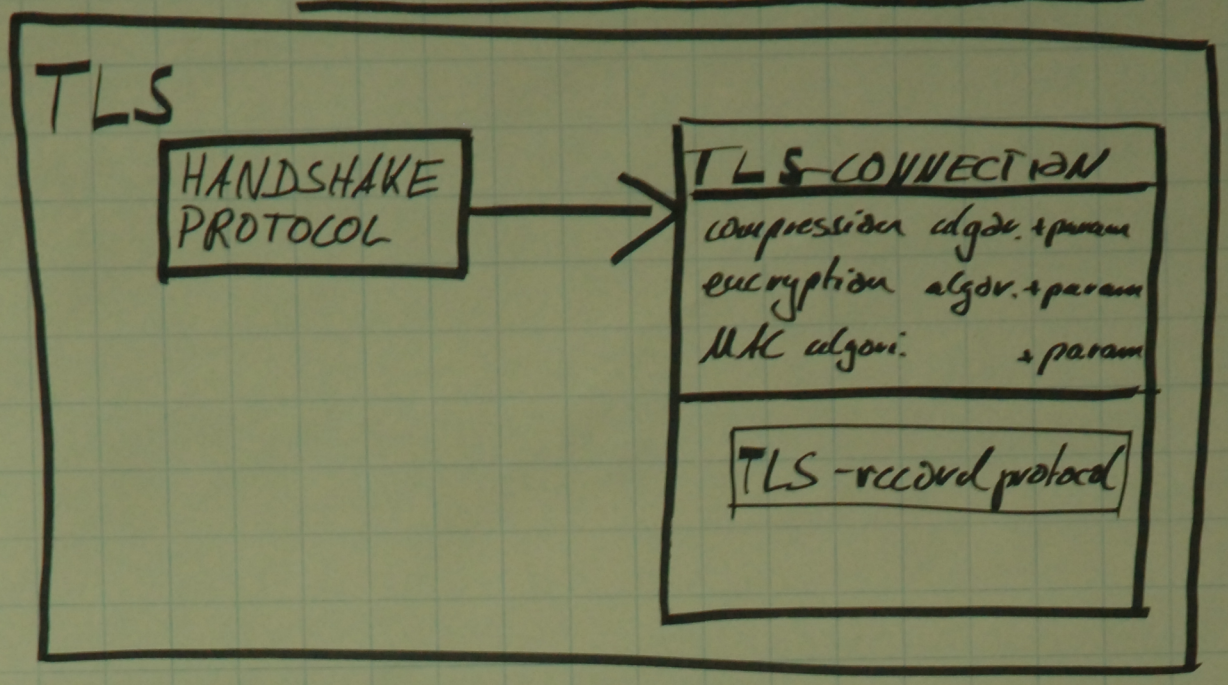
*Certificate
Client Key Exchange
*Certificate Verify
[Change Cipher-Spec
FINISHED →

ENCRYPTED

← [CHANGE CIPHER SPECS]
← FINISHED
← APPLICATION DATA

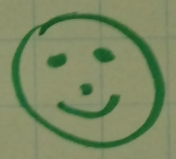
Application data

INFRAESTRUCTURE OF TLS



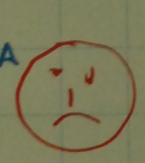
SECURITY CONSIDERATIONS

1º) SESSION KEY AGREEMENT

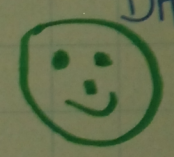


2º) PERFECT FORWARD SECURITY

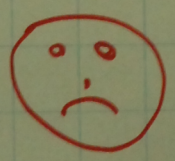
RSA



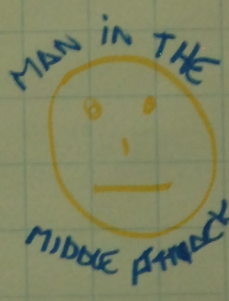
DH



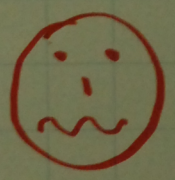
3º) DENIAL OF SERVICE



MAN IN THE MIDDLE ATTACK



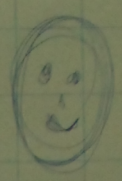
4º) ENDPOINT IDENTIFIER HIDING



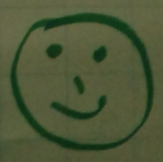
5º) LIVE PARTNER ASSURANCE

6º) PLAUSIBLE DENIABILITY

7º) STREAM PROTECTION



8º) NEGOTIATING PARAMETERS



SSH

- The client initiates the connection.
- Both sides send identification strings.
- Algorithm negotiation (host_key, mac, encryption, compression, etc.).
- Key exchange (output: K - shared secret key; H - exchange hash).
- Service request.
- If needed for the service, the server requests client authentication.
- The communication is now established and proceeds normaly.
- Disconnection.

- Bonus: throughout the communication, key re-exchange might happen.

- Confidentiality: depends on the chosen cipher scheme (DES, AES, etc.).
- Data integrity: if a MAC is computed for each block, integrity provided.
- Replay: the use of unique session identification prevents replay of data from prior sessions.
- Man in the Middle: if used without verifying the association between server_host_key and server_host_name, it's vulnerable, otherwise secure.
- Denial of Service: possible if transport is unreliable because it forces a re-establishment of the connection.
- Perfect Forward Secrecy: the key exchange takes care of it (usually Diffie-Hellman) as long as the session parameters are discarded after use.

Chaum (1985)

19.6.12
es
(1)

Withdrawal protocol

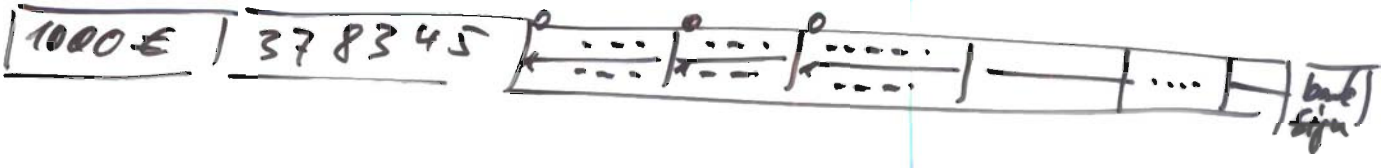
1. Alice prepares 100 anonymous money orders for 1000€ each with a unique string (ie. serial number)

   On each order she adds a list of 73 pairs of identity bit strings, so that xoring a pair gives Alice' identity information. Alice commits to each these 146 bitstrings (secret shares).

   Alice blinds each order and hands them to the bank. →

2. The bank chooses one of the orders and asks Alice to open all the others. ⟷
   The bank then checks all requirements.
   If everything is fine it signs the chosen order blindly and gives it back to Alice ←

3. Alice unblinds and so she has a valid coin:



19.6.12
CS
(2)

Payment protocol

1. Alice spending the coin to Martin
   she hands in the coin. →
2. Martin checks the bank signature.
   If it's wrong: he refuses to accept and calls the Police.
   Then he chooses 73 random bits
   and asks Alice to open one commitment ←
   of each pair according to the random bits
3. Alice does so, →
   Martin checks as far as possible
   and if everything is fine finalizes the bargain.

Deposit protocol

1. Martin gives a complete transcript of
   the payment protocol to the bank.
2. The bank verifies all constraints
   - its signature
   - the uniqueness string
   - the identifying string, and here: the opening to half the secret shares!

   If the signature is wrong: refuse and call Police.
   If the uniqueness string is registered and
   the openings are identical than in the older coin:
   Martin tries to cheat.

If the uniqueness string is registered there are different options:
Alice has tried double spent the coin.
Reconstruct her identity and call the Police.

Properties

- anonymous: The bank cannot identify Alice by the information she gets from the merchant.
- offline: There is no interaction involving the bank during the payment.
- double spending protection: If Alice spends a coin twice she is caught by the identity information.
  Alice cannot frame Martin because Martin chooses the random bits himself.
  Only a coalition of Alice with another (cheating) merchant Mallory may achieve that. To prevent this the merchant may be forced to choose his bits in a special way, eg. by using some for his account number and for date/time.

Chaum, Fiat & Naor (1989)

The coin now is composed of

- basic data: amount, serial #.
- many (eg. 73) tuples (x_i, y_i), $i \in N_{< b/2}$
- bank signature on the previous.

where actually Alice knows values (a_i, b_i, c_i, d_i) such that

$$x_i = g(a_i, c_i)$$
$$y_i = g(b_i, d_i)$$

and

C is a signature on $\prod_{i < b/2} f(x_i, y_i)$

and

$$a_i \oplus b_i = u \| v_i$$

with Alice identifying u.

Questions

What properties must f, g and the signature have?

PROTOCOL. Payment.

1. Alice sends the coin signature C to Martin.
2. Martin chooses some random bit string $z \in \{0,1\}^{k/2}$ and sends it to Alice.
3. Alice computes her answer Z by revealing a half of each identity pair:

$$Z_i = \begin{cases} (a_i, c_i, y_i) & \text{if } z_i = 1, \\ (x_i, b_i, d_i) & \text{if } z_i = 0. \end{cases}$$

   She sends Z to Martin.

4. Martin computes $x_i = g(a_i, c_i)$ or $y_i = g(b_i, d_i)$ according to the value of z_i. He then checks the signature $C^3 = \prod_{i \in \mathbb{N}_{<k/2}} f(x_i, y_i)$ according to (0.3). If everything is OK, he accepts the payment.

$C \longrightarrow$
$\longleftarrow z$
$Z \longrightarrow$

PROTOCOL 0.4. Deposit.

1. Martin sends the entire payment transcript (C, z, Z) to the bank.
2. The bank verifies that the coin is valid and then checks whether the coin has already been deposited by searching for a coin with the same C in her database. If she does not find the coin she puts 100€ on Martin's account and sends him a receipt.
   If, however, she finds a coin (C, z', Z') she detects a double spending. There are two cases:
   - If $z = z'$ and $Z = Z'$ then Martin tries to redeposit an already deposited coin.
   - If $z \neq z'$ then also $Z \neq Z'$ and the bank knows a complete quadruple (a_i, b_i, c_i, d_i) and $a_i \oplus b_i = u||v'$ reveals Alice' identity. The bank calls the police.

   The case $z = z'$ and $Z \neq Z'$ is highly improbable. If transmission errors can be excluded this can only happen if Alice knows a collision for g.

20.6.12
es (2)

Case: Alice tries to forge a coin, i.e. she tries to solve the equations

$$C^3 \equiv_N f(x_0, y_0) \prod_{0<i<\frac{k}{2}} f(x_i, y_i)$$

and $x_i = g(a_i, c_i)$, $y_i = g(b_i, d_i)$

One of her options is to choose $\subset (a_i, b_i, c_i, d_i)$ for $i > 0$ and then try to find (a_0, b_0, c_0, d_0) to fit the remaining equation:

$$f(g(a_0, c_0), g(b_0, d_0)) \equiv_N \underbrace{C^3 \prod_{i>0} f(x_i, y_i)}_{\text{fixed!}}$$

If the bank makes sure that f is one-way then this is a difficult problem.

Case: Alice tries to obtain two coins for the price of one.

Then if Alice has a collision such that

$$f(g(a_0, c_0), g(b_0, d_0)) = f(g(a_0', c_0'), g(b_0', d_0'))$$

with $a_0 \oplus b_0 = a \| b_0$ then she can use this to get a valid which she might spend twice.

20.6.12 es (3)

She chooses further values a_0, b_0, c_0, d_0 from the bank.

Then she spends this coin to some interchange Mallory with a challenge z, say.

Then she spends the coin with a_0', b_0', c_0', d_0' to a cheating merchant Mallory (maybe herself ...) who picks the challenge $z' := (\overline{z_0}, z_1, z_2, \ldots)$.

Since that is a different challenge the bank cannot accuse any merchant, but

$$a_0' \oplus b_0 \quad \text{or} \quad a_0 \oplus b_0'$$

does not reveal Alice' identity and thus the bank cannot accuse her.

Using Mallory for both payments may facilitate the relation z to z' even in the presence of constraints on the challenge.

Of course all this is prevented if we assume that f and g are collision-resistant.

(E)

20.6.12
es (4)

Actually, we assume that:

$f, g : \mathbb{Z}_N \times \mathbb{Z}_N \to \mathbb{Z}_N$.

f, g are both one-way and collision-resistant

and g shall be bijective if its first argument is fixed.

The last requirement assures that (either) x_i (nor y_i) does not convey information about a_i (or b_i), respectively.

Actually, this altogether implies that

$$x_{a_i} = g(a_i, c)$$

is a perfectly hiding and computationally binding commitment to a_i (with randomness c).

For better protection we force the merchant to choose its bits as

h(C, merchant's account #, date/time, ~~Rec~~ bits)

Thus two merchant queries – even with a coalescing merchant – will differ in roughly half of the bits.

To get a coin where this half of commitment pairs reveals a wrong identity, however, Alice & ~~Martin~~ cannot predict which bits will be different. Thus Alice has to send roughly half of the envelopes with $a_i \oplus L_i \neq$ wallet id, say she sends bad envelopes. But the the chance to cheat successfully is

$$\frac{\binom{k-t}{k/2}}{\binom{k}{k/2}} = \left(1-\frac{t}{k}\right)\cdot\ldots\cdot\left(1-\frac{t}{k/2+1}\right) \approx 2^{-t} \leq \left(1-\frac{t}{k}\right)^{k/2} \leq e^{-t/2}$$

So if t is large enough the bank is well-guarded against this type of attack.

We may safely assume $t \geq k/4$ so $e^{-t/2} \leq e^{-k/8}$

exponentially small in k

26.6.12 ex (2)

Seq $t = \varepsilon k$, $k = 128$.

ε	chance of forgery	chance for successful double spending with such a coin
ε	above formula	$2^{-(\frac{1}{2}\cdot\varepsilon)k}$
0	1	2^{-64}
1/8	$2^{-17.54\ldots}$	2^{-48}
1/4	$2^{-39.55\ldots}$	2^{-32}
1/2	$2^{-124.17\ldots}$	1

$\Pi \leq 2^{-64}$

27.6.12 ex (3)

Size of coin: size of N

Size of secret data: $2k$ · size of N

Communication cost: a small multiple of ↺

Efficiency: exponentiation mod N
and a certain number of mult. mod N

→ $O((k + \log N)(\log N)^2)$
+ eval of f, g which should be much cheaper.

Security

- Ok for the bank.

There seems no way for all clients and merchants together to deposit more coins than withdrawn without being detected

Ex How can one find a solution to

$\prod_{i=1}^{t} f(g(a_i, c_i), g(b_i, d_i)) = \prod_{i=1}^{t} f(g(a_i, c_i), g(b_i, d_i))$

PROTOCOL. Withdrawal.

1. Alice chooses $a_i, b_i, c_i, d_i, r_i \in_R \mathbb{Z}_N$ for $i \in \mathbb{N}_{<k}$ at random under condition that

$$a_i \oplus b_i = u||(v+i). \tag{0.1}$$

Here $x \oplus y$ means the binary XOR of the binary representations of the smallest non-negative integers that reduce to x or y. (Write $x = \left(\sum_{j\in\mathbb{N}_{<\lceil\log_2 N\rceil}} x_j 2^j\right) \bmod N$ with $x_j \in \{0,1\}$ such that the integer $\sum_{j\in\mathbb{N}_{<\lceil\log_2 N\rceil}} x_j 2^j$ is less than N. Then use $x_j \oplus y_j$ to define $x \oplus y = \sum_{j\in\mathbb{N}_{<\lceil\log_2 N\rceil}} (x_j \oplus y_j) 2^j$. To avoid difficulties with the allowed range we might require that the topmost bit (the highest significant bit of N) of a_i, b_i and u is always zero.) With $u||(v+i)$ we mean the number with the binary representation $u \cdot 2^{32} + (v+i)$ supposing that we need 32 bits for the counter. Alice computes

$$\begin{aligned} x_i &:= g(a_i, c_i), \\ y_i &:= g(b_i, d_i), \\ B_i &:= r_i^3 f(x_i, y_i) \end{aligned} \tag{0.2}$$

for each i and sends the envelope vector (B_i) to the bank. $\xrightarrow{(B_i)_{i\in\mathbb{N}_{<k}}}$

2. Now the bank chooses a random subset R of $k/2$ indices in $\mathbb{N}_{<k}$ and sends R to Alice. $\xleftarrow{R}$
3. Alice opens the envelopes chosen by R by sending $(a_i, b_i, c_i, d_i, r_i)_{i\in R}$ to the bank. $\xrightarrow{(a_i,\dots)_{i\in R}}$
4. The bank tests (0.1) and (0.2) for $i \in R$. If this turns out well she computes

$$s := \prod_{i\notin R} B_i^{(1/3)}$$

and sends s to Alice. The bank charges Alice's account 100 € and increments the counter v by k. $\xleftarrow{s}$

5. Alice can then easily unblind this signature and obtains $C = s/\prod_{i\notin R} r_i$. She re-indexes the identity pairs (a_i, b_i, c_i, d_i) to the indices $0, \dots, k/2-1$ such that $\tilde{f}(x_0, y_0) < \tilde{f}(x_1, y_1) < \cdots < \tilde{f}(x_{k/2-1}, y_{k/2-1})$. Finally, Alice increments her copy of v by k. Now she has a coin

$$\left((a_i, b_i, c_i, d_i)_{i\in\mathbb{N}_{<k/2}}, C\right)$$

which fulfills the condition that

$$\begin{aligned} a_i \oplus b_i &= u||V(i), \\ x_i &= g(a_i, c_i), \\ y_i &= g(b_i, d_i), \\ C^3 &= \prod_{i\in\mathbb{N}_{<k/2}} f(x_i, y_i). \end{aligned} \tag{0.3}$$

- Ok for the merchant?

  Yes, if the "random bits", the challenge of the merchant is well-constructed then there is no way to double deposit a coin even if all other stuff is just simulated...
  We just have to make sure that the challenge construction is collision-resistant or even collision-free.

- Ok for the client?

  No: the bank can frame Alice for double-spending by simulating all players since there is no secret of Alice involved.

  In turn that means that the bank never has a proof for double spending!

  → Bad for the bank!

Need to add in a proof that it was Alice who ~~with~~ has withdrawn the coin.
To avoid this problem replace $(0,1)$ by

$$c_i \oplus b_i = c \,||\, (v+i) \,||\, sig_{Alice}(v||v+i||s_i)$$

21.6.12
es
(3)

Historical remarks

Several ecash systems have been tried to make real:

- DigiCash (David Chaum)
- eCash / Cybercash (Hettinga)
- floozy
- ...

As far as I know, none of systems were or are used in practice.

Chaum, that's Naor is essentially based on the security of RSA.

Other constructions followed.

Eg: Brands (1993) suggested various options all based on the

Representation Problem in group G

Given (publically) elements $g_1, \dots, g_r \in G$.

The problem is to find

given $h \in \langle g_1, \dots, g_r \rangle \subseteq G$

find a representation $h = g_1^{a_1} \cdot \dots \cdot g_r^{a_r}$.

typically: $G = \langle g_i \rangle$

27.6.12 es (1)

This problem is closely related to the DLP.

Brands actually many variants with additional properties

- with observers
- fair, transferable, ...
- "crime protection"

3.7.12 es (1)

Complexity of Chaum, Fiat & Naor

- We work with RSA: k_1-bit modulus N.
- We need a certain # of secret shares: k.

Factoring RSA 703 : 2012 ≈ 500 core years.

Typical: $k_1 = 2048$, $k = 256$.

bits communicated Alice – Bank ≈ $\frac{7}{2} k \cdot k_1 + k_1$

Alice – Merkin ≈ $k_1 + \frac{3}{2} k \cdot k_1$

Merkin – Bank

size of coin public data: k_1

secret data: $2k$ k_1

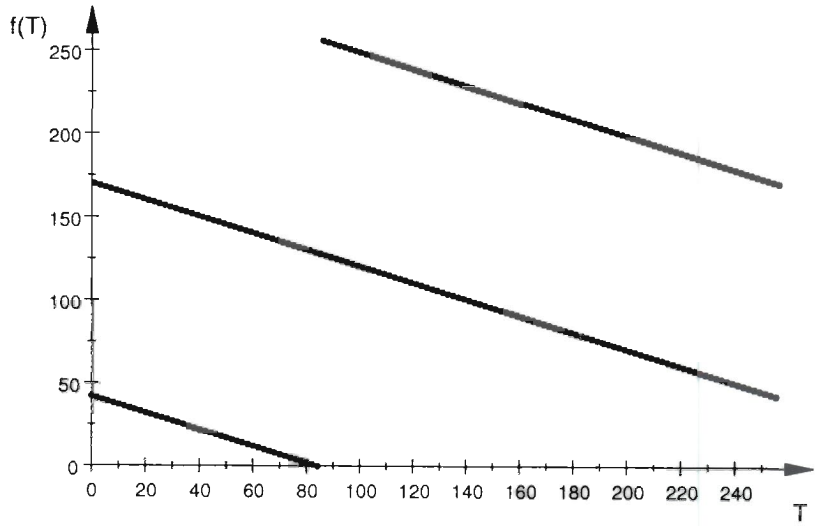


Figure 1: The line $f\colon \mathbb{F}_{257} \to \mathbb{F}_{257},\ T \mapsto 128\,T + 42$ over the field $\mathbb{F}_{257}$ carries the secret $f(0) \mathrel{\widehat{=}} 42$ and passes through zero at $T \mathrel{\widehat{=}} 84$. The elements of $\mathbb{F}_{257}$ are represented as integers modulo 257 (which is prime!).

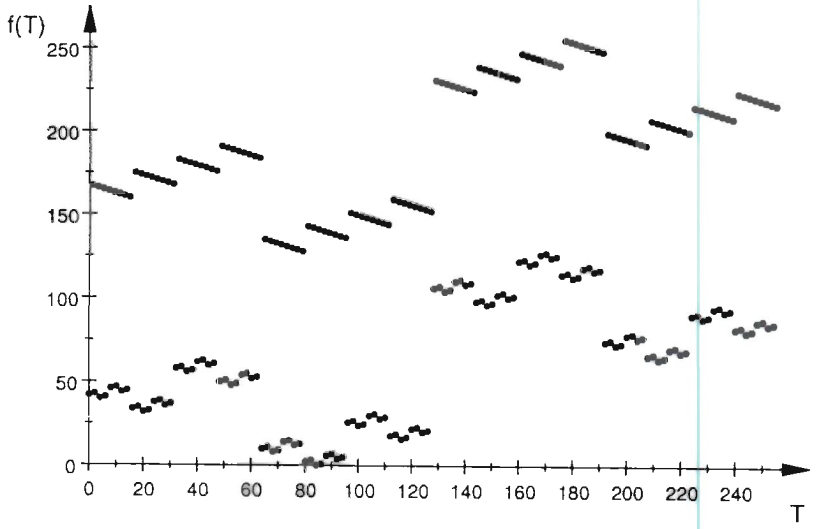


Figure 2: The line $f\colon \mathbb{F}_{256} \to \mathbb{F}_{256},\ T \mapsto (x^7 + x^3 + x^2 + 1)\,T + (x^5 + x^3 + x)$ over the field $\mathbb{F}_{256}$ carries the secret $f(0) = x^5 + x^3 + x \mathrel{\widehat{=}} 2^5 + 2^3 + 2 = 42$ and passes through zero at $T \mathrel{\widehat{=}} 84$. The elements of $\mathbb{F}_{256}$ are represented as polynomials in x of degree less than 8 over $\mathbb{F}_2 = \mathbb{Z}_2$ modulo $x^8 + x^4 + x^3 + x + 1$ and identified with integers by 'evaluating' such a polynomial over the integers at $x = 2$.

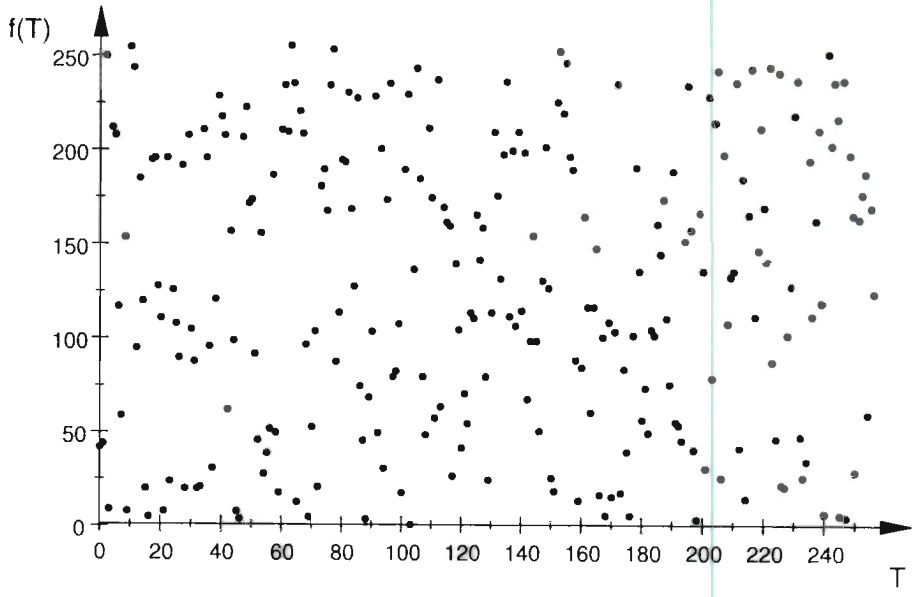


Figure 3: The cubic curve $f\colon \mathbb{F}_{257} \to \mathbb{F}_{257},\ T \mapsto 20\,T^3 + 42\,T^2 + (-60)\,T + 42$ over $\mathbb{F}_{257}$ on the right hand side each carries the secret $f(0) \mathrel{\widehat{=}} 42$. For our untrained eyes the nice structure of this curve is not visible but still: any four points determine the entire polynomial and thus the secret.

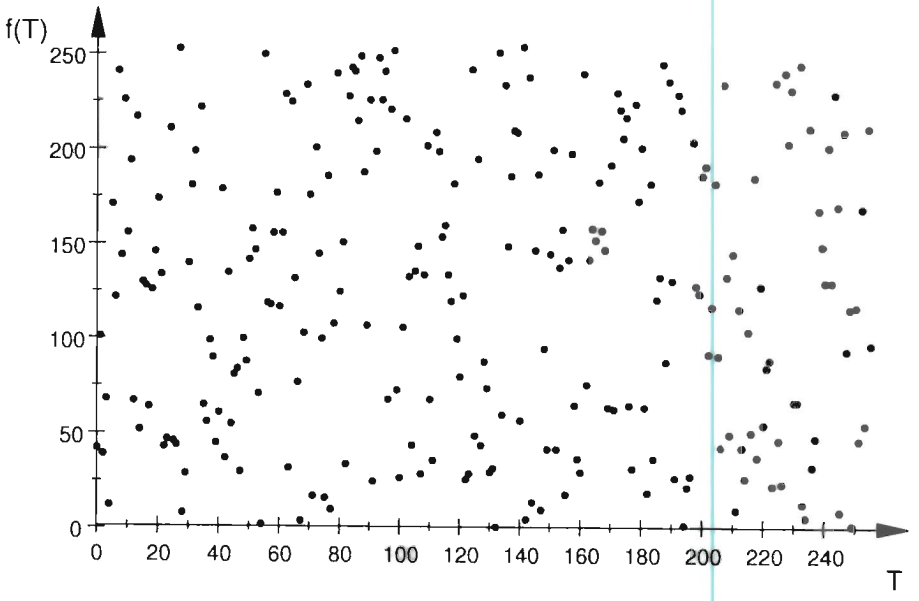
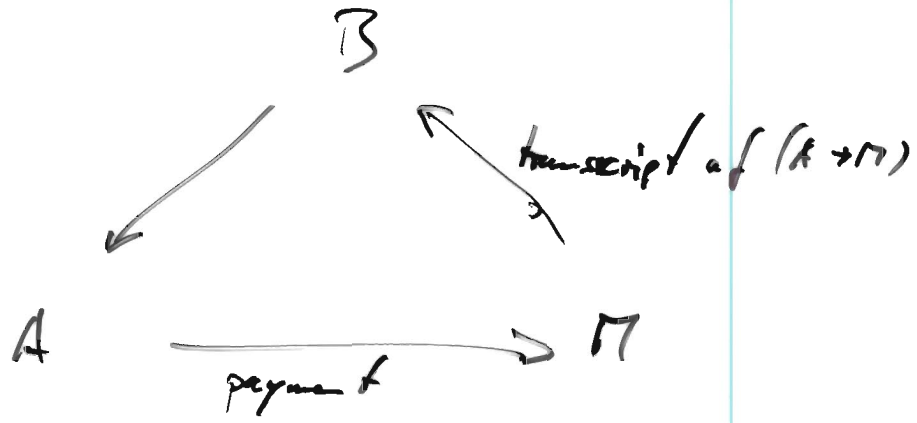


Figure 4: The cubic curve $f\colon \mathbb{F}_{256} \to \mathbb{F}_{256},\ T \mapsto (x^7 + x^6 + x^5 + x^4 + x^3 + x + 1)\,T^3 + (x^7 + x^5 + x + 1)\,T^2 + (x^4 + x^2 + x + 1)\,T + (x^5 + x^3 + x)$ over $\mathbb{F}_{256}$ carries the secret $f(0) \mathrel{\widehat{=}} 42$. For our untrained eyes the nice structure of this curve is not visible but still: any four points determine the entire polynomial and thus the secret.

31.7.12
cs



Requirements:

- Martin's challenge must be large to be suitably unpredictable / manipulable?
  → Need exponentially many possible challenges
- Double-spending protection.
  → Need any two different challenges must reveal the secret.
- Each challenge must be answerable only with some information created by the bank using her secrets.

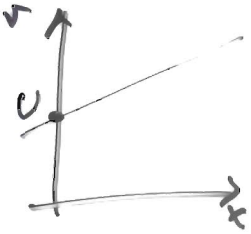
Idea:

Use secret sharing schemes that are more general than the perfect scheme used in Chaum, Fiat & Naor.

We could for example fix a line

$r = kx + U$ over some field $\mathbb{F}$

where k is a randomly chosen value in $\mathbb{F}$ and U is an element of $\mathbb{F}$ encoding Alice's identity



The Bank's challenge could be a value $x \neq 0$ and Alice's reply would be the corresponding r.

Clearly: as long as Alice spends the coin only once the pair (x, r) tells nothing about U.

Because there is always a (unique) line through (x, r) and $(0, U)$...

But if Alice spends the coin twice the Bank obtains (x_1, r_1) and (x_2, r_2) both on the line.

Now the system

$$r_1 = k \cdot x_1 + U$$
$$r_2 = k \cdot x_2 + U$$

determines U (and k) if only $x_1 \neq x_2$.

But how to bind Alice to her answer? Notice that Alice does not want to reveal k and U!

The core of Fergusan's solution uses those group elements A, B, C, and bank signatures of RSA type

$$AC^k = S_1^v \quad \text{in } \mathbb{Z}_N$$

and

$$BC^U = S_2^v \quad \text{in } \mathbb{Z}_N$$

where (N, v) is public RSA key of the bank.

Observe that

$$(S_1^x S_2)^v = A^x C^{kx} \cdot BC^U = A^x B C^{kx+U} = A^x B C^r .$$

and so it proves that (x, r) is a point on the above because $S = S_1^x S_2$ is a bank signature on $A^x B C^r$.

This is not completely true if

$$r \neq \varphi(N) kx + U .$$

We need to make sure that signature S and $A^x B C^r$ fit together without any tweak on Martin's side.

Clearly, if $k, x, v \in \mathbb{Z}$ then by the above we have

$$(S_1^x S_2)^v = A^{x\,kx+u} B C .$$

Now Alice writes $kx+u = r + \hat{r} \cdot v$.

Then

$$A^x B C^r = A^x B C^{kx+u-\hat{r}v}$$

$$= (S_1^x S_2)^v \cdot C^{-\hat{r}v}$$

$$= \underbrace{\left(S_1^x S_2 C^{-\hat{r}}\right)}_{=: S}{}^v$$

This is what Ferguson makes work!

PROTOCOL 1. Bank setup.

1. The bank chooses an RSA key pair: Find p, q primes such that $N = p \cdot q$ has, say, 1024 bits. Compute $L = (p-1)(q-1)$ and choose a suitably large, say 128 bit, prime $v \in \mathbb{N}_{<L}$ coprime to L. Calculate $(1/v) = v^{-1} \bmod L$.
2. Find elements $g_1, g_2, g_3 \in \mathbb{Z}_N^\times$ of large order. This might be a problem since the bank must know the factorization of L to determine the order of a randomly chosen element. But p and q can be constructed such that at least a large prime factor P of $p-1$ and Q of $q-1$ is known. Then $x^{\frac{L}{PQ}}$ is an element of order 1, P, Q, or PQ for any $x \in \mathbb{Z}_N^\times$ and elements of order PQ can be found by repeating this some times.
3. Find a prime t with $t \equiv_N 1$. (Since the size of t needs only be of the same order as N, she might simply choose the smallest prime of the form $xN+1$.)
4. Find elements $h_2, h_3 \in \mathbb{F}_t^\times$ of order N. Again this is easy by testing the order of $x^{\frac{t-1}{N}}$ for randomly chosen $x \in \mathbb{F}_t^\times$ which can be only 1, p, q, or $N = pq$.
5. Fix hash functions $f_1 \colon \mathbb{Z}_N^\times \to \mathbb{N}_{<v}$, $f_2, f_3 \colon \mathbb{F}_t^\times \to \mathbb{N}_{<v}$, and $f_4 \colon \mathbb{N}_{<v} \times \mathbb{N}_{<v} \to \mathbb{N}_{<v}$.
6. Publish t
$$(N, v, g_1, g_2, g_3, h_2, h_3, f_1, f_2, f_3, f_4).$$
The only extra information the bank needs is its secret exponent $(1/v)$.

Fix a way to compute

$A = F_1(a) = a g_1^{f_1(a)}$

$B = F_2(b)$

$C = F_3(c)$

based on the fixed hash functions.

PROTOCOL 2. Payment.

1. Alice hands over (a, b, c) to Martin. $\xrightarrow{(a,b,c)}$
2. Martin chooses a random challenge $x \in \mathbb{N}_{<v}$. $\xleftarrow{x}$
3. Alice computes $r + \widehat{r}v \leftarrow kx + U$ with $r \in \mathbb{N}_{<v}$ and a signature R to $A^x BC^r$ by $R \leftarrow S_1^x S_2 C^{-\widehat{r}} = (A^x BC^r)^{(1/v)}$. She sends (r, R) to Martin. $\xrightarrow{(r,R)}$
4. Martin verifies that the signature is valid: all transmitted data are in the required domains and

$$R^v \stackrel{?}{=} A^x BC^r.$$

Note that he can do that.

PROTOCOL 3. Deposit.

1. Martin sends the entire transcript of the payment Protocol 2 to the bank.
2. She then looks up the ~~signature~~ in her database.

   — coin (a, b, c)

   - If she does not find it, Martin gets his money put on his account and a receipt.
   - Otherwise, the bank detects a double spending just as in the other systems:
     - If the challenges x and x' are also equal then Martin has tried to redeposit a coin.
     - Otherwise the bank tries to reveal Alice' identity. For now the bank knows $r \equiv_v kx + U$ and $r' \equiv_v kx' + U$ modulo v which is just a linear system of equations for k and U. Now she can take Alice to court for double spending.

4.7.12
es ①

How to generate blind signatures to a random quantity?

What do we need?

One thing: we should avoid the possibility that Alice knows discrete logs between A, B and C. Because for example if she knows that $A = C^{\delta}$.

Then from $AC^{k} = S_1^{v}$

she obtains (#) $C^{k+\delta} = S_1^{v}$

which she may use to modify her reply (r, R) obtained legally and solving

(*) $A^{x} B C_{\alpha}^{r} = R^{v}$

(multiplying (*) with (#)$^{\alpha}$ we get

$$A^{x} B C^{r+\alpha(k+\delta)-\beta v} = (R S_1^{\alpha} C^{-\beta})^{v}$$

where Alice may choose α freely and adapts β such that $r + \alpha(k+\delta) - \beta v \in \mathbb{N}_{<v}$.

So we must make sure that Alice does not have the power to choose A, B, C during the withdrawal to her own liking. But still we want that A, B, C are verifiably random and unknown to the bank.

4.7.12
es
(2)

You have studied coin flipping by phone...

We want $a \in \mathbb{Z}_N^\times$ to be chosen at random. This can be accomplished if Alice picks $a' \in \mathbb{Z}_N^\times$ and the bank picks $a'' \in \mathbb{Z}_N^\times$ and the

$$a = a' a''.$$

is uniformly random over $\mathbb{Z}_N^\times$ if only one of a' and a'' is chosen uniformly random.

So we need that the first party chooses and then commits to its choice without revealing it. Then the second party can safely reveal her number and go along.

Additional problem: the bank must never know the answer!

Let's look at a solution how to obtain a random a and a signature on $A = a\, g^{f(a)}$ (where $g = g_1$, $f = f_1$ from the bank setup) such that a is unpredictable for the bank but she still be sure that her part a'' has been used.

PROTOCOL 4. Randomized blind signature.

1. Alice randomly chooses $a', \alpha \in \mathbb{Z}_N^\times$ and $\sigma \in \mathbb{N}_{<v}$. She computes $\widetilde{A} \leftarrow \alpha^v a' g^\sigma$ and sends that to the bank. $\xrightarrow{\widetilde{A}}$
2. The bank randomly chooses $a'' \in \mathbb{Z}_N^\times$ and sends it to Alice. $\xleftarrow{a''}$
3. Alice computes $a \leftarrow a'a'' \in \mathbb{Z}_N^\times$ and an adjusting exponent $e + \widehat{e}v \leftarrow f(a) - \sigma$ with $e \in \mathbb{N}_{<v}$ and sends e to the bank. $\xrightarrow{e}$
4. The bank computes $\overline{A} \leftarrow \widetilde{A} \cdot a'' g^e$ and sends Alice a signature $\widetilde{S} \leftarrow \overline{A}^{(1/v)}$ of it. $\xleftarrow{\widetilde{S}}$
5. Alice unblinds the signature to obtain $S \leftarrow \widetilde{S}\alpha^{-1} g^{\widehat{e}}$. Now she has a signature pair (a, S) satisfying

$$(*) \qquad S^v \stackrel{?}{=} a g^{f(a)}.$$

So

$$\begin{aligned}
S^v &= (\tilde{S}\alpha^{-1} g^{\hat{e}})^v \\
&= \tilde{S}^v \alpha^{-v} g^{\hat{e}v} \\
&= \overline{A}\, \alpha^{-v} g^{\hat{e}v} \\
&= \widetilde{A}\, a'' g^e \alpha^{-v} g^{\hat{e}v} \\
&= \alpha^v a' g^\sigma a'' g^e \alpha^{-v} g^{\hat{e}v} \\
&= a\, g^{\sigma + \underbrace{e + \hat{e}v}_{f(a)-\sigma}} \\
&= a\, g^{f(a)}.
\end{aligned}$$

10.7.12
es. (4)

Relation in Protocol 4:

The honest bank makes sure that

(*) $\hat{S}^{v} = \hat{A} \cdot a''^{} g^{e}$.

But this also follows from Alice's result

$S^{v} = a^{f(a)} g$ ...

Actually, it is the only relation!

If (*) holds for a transcript $(\hat{A}, a'', e, \hat{S})$ then given any a there is a unique choice for the remaining parameters that can have produced this:

$\hat{A} = \alpha^{v} a'' g^{\sigma}$ ← fixes α.

a'' : $a = a' a''$ ← fixes a'

e : $e + \hat{e} v = f(a) - \sigma$ ← fixes σ

$\hat{S}$: $\hat{S}^{v} = \hat{A}\, a''^{} g^{e}$ ← checks (*)

Seen from the bank's side: every a has the same probability. ⇒ Anonymity / Blindness

Since $a = a' a''$ the final outcome is ~~random~~ uniformly random if at least only one party chooses its share uniformly at

PROTOCOL 5. Randomized blind signature without exponential attack.

1. Alice randomly chooses $a', \alpha \in \mathbb{Z}_N^\star$ and $\sigma \in \mathbb{N}_{<v}$. She computes $\widetilde{A} \leftarrow \alpha^v a' g^\sigma$ and sends that to the bank. $\xrightarrow{\widetilde{A}}$
2. The bank randomly chooses $a'' \in \mathbb{Z}_N^\times$, computes $\widetilde{h} \leftarrow h^{a''}$ and sends it to Alice. $\xleftarrow{\widetilde{h}}$
3. Alice computes an adjusting exponent $e + \widehat{e}v \leftarrow f(\widetilde{h}^{a'}) - \sigma$ with $e \in \mathbb{N}_{<v}$ and sends it to the bank. $\xrightarrow{e}$
4. The bank computes $\overline{A} \leftarrow \widetilde{A} \cdot a'' g^e$ and sends Alice a signature $\widetilde{S} \leftarrow \overline{A}^{(1/v)}$ of it along with a''. $\xleftarrow{a'', \widetilde{S}}$
5. Alice calculates $a \leftarrow a'a''$ and unblinds the signature to obtain $S \leftarrow \widetilde{S}\alpha^{-1} g^{\widehat{e}}$. Now she has a signature pair (a, S) satisfying

(**) $$S^v \stackrel{?}{=} a g^{f(h^a)}.$$

Clear : Correct ✓
Efficient
Secure? Reduction? :(

PROTOCOL 6. Withdrawal.

1. Alice chooses random shares $a', b', c' \in_R \mathbb{Z}_N^\times$, random blinding bases $\alpha, \beta, \gamma \in_R Z_N^\times$, and random blinding exponents $\sigma, \tau, \varphi \in_R \mathbb{N}_{<v}$. She computes the blinded candidates $\widetilde{A} \leftarrow \alpha^v a' \cdot g_1^\sigma$, $\widetilde{B} \leftarrow \beta^v b' \cdot g_2^\tau$, $\widetilde{C} \leftarrow \gamma^v c' \cdot g_3^\varphi$ and sends them to the bank. $\xrightarrow{\widetilde{A}, \widetilde{B}, \widetilde{C}}$
2. The bank chooses her random shares $a'', b'', c'' \in_R \mathbb{Z}_N^\times$ and sends a'', $\widetilde{h_2} \leftarrow h_2^{b''}$, $\widetilde{h_3} \leftarrow h_3^{c''}$ to Alice. $\xleftarrow{a'', \widetilde{h_2}, \widetilde{h_3}}$
3. Alice computes

$$e_2 + \widehat{e_2} v \leftarrow f_2(\widetilde{h_2}^{b'}) - \tau \qquad \text{with } e_2 \in \mathbb{N}_{<v},$$
$$e_3 + \widehat{e_3} v \leftarrow f_3(\widetilde{h_3}^{c'}) - \varphi \qquad \text{with } e_3 \in \mathbb{N}_{<v}.$$

and chooses $k' \in_R \mathbb{N}_{<v}^\times$. After computing $a \leftarrow (a'a'' \cdot f_4(e_2, e_3))^{k'}$ and $k^- \in \mathbb{N}_{<v}$, $\widehat{1} \in \mathbb{N}$ such that $k'k^- = 1 + \widehat{1}v$, she computes

$$e_1 + \widehat{e_1} v \leftarrow k^- f_1(a) - \sigma \qquad \text{with } e_1 \in \mathbb{N}_{<v}.$$

Then she sends the exponents (e_1, e_2, e_3) to the bank. $\xrightarrow{e_1, e_2, e_3}$
4. The bank computes $\overline{A} \leftarrow \widetilde{A} a'' f_4(e_2, e_3) g_1^{e_1}$, $\overline{B} \leftarrow \widetilde{B} b'' g_2^{e_2}$, $\overline{C} \leftarrow \widetilde{C} c'' g_3^{e_3}$. Then the bank chooses her share $k'' \in_R \mathbb{N}_{<v}^\times$ of k. She then computes the signatures $\widetilde{S}_1 \leftarrow (\overline{A}\,\overline{C}^{k''})^{(1/v)}$ and $\widetilde{S}_2 \leftarrow (\overline{B}\,\overline{C}^U)^{(1/v)}$ and sends them to Alice. $\xleftarrow{b'', c'', k'', \widetilde{S}_1, \widetilde{S}_2}$
5. Alice puts everything together: she computes $b \leftarrow b'b''$, $c \leftarrow c'c''$ in $\mathbb{Z}_N^\times$, and $k + \widehat{k}v \leftarrow k'k''$ with $k \in \mathbb{N}_{<v}$. Now she can compute

$$A \leftarrow a g_1^{f_1(a)}, \quad B \leftarrow b g_2^{f_2(h_2^b)}, \quad C \leftarrow c g_3^{f_3(h_3^c)}$$

and unblind the signatures $S_1 \leftarrow \left(\widetilde{S}_1 \left(\alpha^{-1} g_1^{\widehat{e_1}}\right) \left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^{k''}\right)^{k'} g_1^{-f_1(a)\widehat{1}} C^{-\widehat{k}}$ and $S_2 \leftarrow \widetilde{S}_2 \left(\beta^{-1} g_2^{\widehat{e_2}}\right) \left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^U$. She now has a coin (a, b, c, k, S_1, S_2) with the property

$$(***) \qquad S_1^v = AC^k, \quad S_2^v = BC^U.$$

$$\bar{A}^{k'} = \left(\tilde{A}\, a^{\mu}\, f_{\mu}(e_2, e_3)\, g_1^{\,e_1} \right)^{k'}$$

$$\tilde{A} = \alpha^{\nu}\, a'\, g_1^{\,\sigma}$$

$$= \left(\alpha^{\nu} \underline{a'}\, g_1^{\,\sigma}\, \underline{a^{\mu}}\, \underline{f_{\mu}(e_2, e_3)}\, g_1^{\,e_1} \right)^{k'}$$

$$A = a\, g_1^{\,f_1(a)}$$

$$a = \left(a' a^{\mu} f_{\mu}(e_2, e_3) \right)^{k'}$$

$$= a \left(\alpha^{\nu} g_1^{\,\sigma + e_1} \right)^{k'}$$

$$e_1 + \hat{e}_1 \nu = k\, f_1(a) - \sigma$$

$$= a \left(\alpha^{\nu} g_1^{-\hat{e}_1 \nu} \right)^{k'} g_1^{\,k\, f_1(a)}$$

$$k\, k' = 1 + \hat{f}\, \nu$$

$$= a\, g_1^{\,f_1(a)}\, \alpha^{\nu k'}\, g_1^{-\hat{e}_1 \nu k'}\, g_1^{\,f_1(a)\, \hat{f}\, \nu}$$

$$\bar{A}^{k'} = A \cdot \left(\left(\alpha\, g_1^{-\hat{e}_1} \right)^{k'} g_1^{\,f_1(a)\, \hat{f}} \right)^{\nu}$$

$$\bar{B} = B \cdot \left(\beta\, g_2^{-\hat{e}_2} \right)^{\nu}$$

$$\bar{C} = C \cdot \left(\gamma\, g_3^{-\hat{e}_3} \right)^{\nu}$$

11.7.12
es (4)

So let's check $S_1^{\vee} = A C^k$:

$$S_1^{\vee} = \left[\left(\tilde{S}_1 \, (\alpha^{-1} g_1^{\vec{e}_1}) \, (\delta^{-1} g_3^{\hat{e}_3})^{k''} \right)^{k'} g_1^{-f_1(a)\vec{r}} \, C^{-\hat{k}} \right]^{\vee}$$

$$= \bar{A}^{k'} \left((\alpha^{-1} g_1^{\hat{e}_1})^{k'} g_1^{-f_1(a)\vec{r}} \right)^{\vee} \cdot \left(\bar{C} \, (\delta^{-1} g_3^{\hat{e}_3})^{\vee} \right)^{k'k''} \cdot C^{-\hat{k}^{\vee}}$$

$\tilde{S}_1^{\vee k'} = \bar{A}^{k'} \bar{C}^{k''k'}$

$$= A \cdot C^{k'k'' - \hat{k}^{\vee}}$$

$k'k'' = k + \hat{k}^{\vee}$

$$= A \cdot C^{k}.$$

Similarly, we find that $S_2^{\vee} = B \cdot C^{u}$.

Bottom line: The method used is correct.

Efficiency: ok ✓

Last question: security?

1. Anonymity?

Same reasoning as with the original simple randomized blind signature scheme: assume a transcript is

11.7.12
ES

given, choose a, b, c, k and try to solve for the hidden parameters. Observe that there is exactly same number of solutions in each case.

2. Unforgeability?

We just note that all our trials run into well-known extractions and/or discrete logarithm problems. (or combinations).

We now have:

- probably unforgeable coins
- anonymous
- double spending protected
- very efficient wrt. coin size

Framing protection for Alice has to be added. It cannot be simply embedded into O since the bank could skip the withdrawal and continue without Alice or similar

5. Solutions for electronic money

Chaum. Brands. Further systems for additional properties like divisibility or smart card 'agencies'.

5.1. Chaum's system. See Schneier (1996), section 6.4, pages 139ff.

PROTOCOL 5.1. Chaum's electronic cash.

1. Alice prepares 100 anonymous money orders for $1\,000$€ each with a uniqueness string (that is, a serial number). On each order she adds a list of 73 pairs of identity bit strings, so that xoring a pair gives Alice' identity information, her name and account number, say. Alice commits to each of these 146 bit strings. Alice blinds each order and hands them to the bank.
2. The bank asks Alice to open all but one including all the identity string pairs and checks whether all data are as required. If so it signs the remaining order blindly.
3. Now Alice has a valid coin.

4. For spending it to the merchant Martin, she hands him the coin.
5. Martin verifies the bank's signature. If it is wrong he refuses to accept and calls the Police.
6. Then he chooses 73 random bits and asks Alice to open the left or right half of the identity pairs accordingly.
7. Alice does so.

8. Martin takes the money to the bank.
9. The bank verifies all constraints:
   - its signature,
   - the uniqueness string,
   - the identity strings.

   If the signature is wrong the bank refuses and calls the Police.
   If the uniqueness string is registered and the identity strings are opened as in the earlier coin, Martin tries to cheat.
   If the uniqueness string is registered and the identity strings are differently opened, the bank reconstructs Alice' identity information and calls the Police.

This system is *anonymous*: the bank cannot identify Alice by the information she gets from the merchant. And it is *offline*: It does not require an interaction with the bank during the payment. There is a protection against *double spending*: If Alice spends a coin twice she is caught by the identifier. If the bank detects a double spending it can distinguish whether Martin or Alice tried to cheat: If the bank receives a coin with the same uniqueness string but different identity strings then Alice tried to cheat. If also the identity strings are equal then Martin has to be blamed. Also Alice cannot frame Martin since she cannot control how Martin chooses his challenge. Only an alliance of Alice and Mallory, another merchant, may achieve this: if Mallory simply asks the same challenge as Martin and he is faster to be at the bank then ... To prevent this last kind of fraud the merchant can be forced to use special challenges that depend on his account number and the present date and time.

5.2. Chaum, Fiat & Naor (1989). Chaum's original cut-and-choose variant has the major disadvantage that the probability of cheating by using a faked envelope is still $\frac{1}{n}$ if n envelopes are used. Further, the last section does not yet tell us how to hide the identity information in a practical implementation such that Alice is really bound to it.

The following set of protocols gives an answer to these questions. We assume that the bank has published its public RSA key $(N, 3)$ and a security parameter k (specifying the number of 'envelopes' and 'identity splits' to use). For simplicity, let k be a multiple of 4. And the bank has fixed two collision-resistant functions $f, g\colon \mathbb{Z}_N \times \mathbb{Z}_N \to \mathbb{Z}_N$ and such that g with any fixed first argument gives a one-to-one (bijective) map. Further, Alice has opened the account number u and obtained a counter v that has to be advanced in every withdrawal.

PROTOCOL 5.2.

1. Alice chooses $a_i, b_i, c_i, d_i, r_i \in_R \mathbb{Z}_N$ for $i \in \mathbb{N}_{<k}$ at random under condition that

$$a_i \oplus b_i = u||(v+i). \tag{5.3}$$

Here $x \oplus y$ means the binary XOR of the binary representations of the smallest non-negative integers that reduce to x or y. (Write $x = \left(\sum_{j\in\mathbb{N}_{<\lceil\log_2 N\rceil}} x_j 2^j\right) \bmod N$ with $x_j \in \{0,1\}$ such that the integer $\sum_{j\in\mathbb{N}_{<\lceil\log_2 N\rceil}} x_j 2^j$ is less than N. Then use $x_j \oplus y_j$ to define $x \oplus y = \sum_{j\in\mathbb{N}_{<\lceil\log_2 N\rceil}} (x_j \oplus y_j) 2^j$. To avoid difficulties with the allowed range we might require that the topmost bit (the highest significant bit of N) of a_i, b_i and u is always zero.) With $u||(v+i)$ we mean the number with the binary representation $u \cdot 2^{32} + (v+i)$ supposing that we need 32 bits for the counter. Alice computes

$$\begin{aligned} x_i &:= g(a_i, c_i), \\ y_i &:= g(b_i, d_i), \\ B_i &:= r_i^3 f(x_i, y_i) \end{aligned} \tag{5.4}$$

for each i and sends the envelope vector (B_i) to the bank. $\xrightarrow{(B_i)_{i\in\mathbb{N}_{<k}}}$

2. Now the bank chooses a random subset R of $k/2$ indices in $\mathbb{N}_{<k}$ and sends R to Alice. $\xleftarrow{R}$
3. Alice opens the envelopes chosen by R by sending $(a_i, b_i, c_i, d_i, r_i)_{i\in R}$ to the bank. $\xrightarrow{(a_i,\dots)_{i\in R}}$
4. The bank tests (5.3) and (5.4) for $i \in R$. If this turns out well she computes

$$s := \prod_{i\notin R} B_i^{(1/3)}$$

and sends s to Alice. The bank charges Alice's account 100 € and increments the counter v by k. $\xleftarrow{s}$

5. Alice can then easily unblind this signature and obtains $C = s/\prod_{i\notin R} r_i$. She re-indexes the identity pairs (a_i, b_i, c_i, d_i) to the indices $0, \dots, k/2-1$ such that $\check{f}(x_0, y_0) < \check{f}(x_1, y_1) < \cdots < \check{f}(x_{k/2-1}, y_{k/2-1})$. Finally, Alice increments her copy of v by k. Now she has a coin

$$\Big((a_i, b_i, c_i, d_i)_{i\in\mathbb{N}_{<k/2}}, C\Big)$$

which fulfills the condition that

$$\begin{aligned} a_i \oplus b_i &= u||V(i), \\ x_i &= g(a_i, c_i), \\ y_i &= g(b_i, d_i), \\ C^3 &= \prod_{i\in\mathbb{N}_{<k/2}} f(x_i, y_i). \end{aligned} \tag{5.5}$$

To withdraw a coin with invalid identity information Alice would have to send some wrong B_i to the bank. But if she does so in only ε of all her envelopes then the chance of not being caught is $\binom{k(1-\varepsilon)}{k/2} / \binom{k}{k/2}$, which is at most $(1-\varepsilon)^{k/2} \le \mathrm{e}^{-\frac{1}{2}\varepsilon k}$. On the other hand to hide her identity in double spending she must get a pair of challenges that differs only on that ε proportion, thus the chance of getting a challenge that allows her to re-spend a coin is at most $\left(\frac{1}{2}\right)^{(1/2-\varepsilon)k}$. With $k = 128$ we get the following probabilities:

ε	Chance of forgery	Chance for successful double spending
0	1	2^{-64}
$1/8$	$2^{-17.54\dots}$	2^{-48}
$1/4$	$2^{-39.55\dots}$	2^{-32}
$1/2$	$2^{-124.17\dots}$	1

To forge a coin without cheating the bank Alice would have to produce such a set of information fulfilling (5.5). If Alice can forge bank signatures that is no problem, she just computes a valid C.

Otherwise she must adapt the right hand side to give a cube with a known root. Say, as a particular case, she chooses all but a_0, b_0, c_0, d_0 in advance. Then the remaining task is to find these to give a particular value for $f(g(a_0, c_0), g(b_0, d_0))$. But that means that Alice has an efficient way to compute preimages of that combination of f and g and therefore of f. That would mean that f is not one-way and thus not collision-resistant.

A third possibility would be to withdraw a valid coin but later use different values for the coin values that do not reveal the true identity but some garbage. Alice could do that if she knew collisions for g. Suppose $g(x, y) = g(x', y')$ with $x \neq x'$ is one such collision. Then she uses $a_0 = x$, $c_0 = y$ with the bank but when she spends the coin she uses $a_0 = x'$, $c_0 = y'$. Yet, finding collisions for g is supposed to be difficult. Of course, Alice has more room for variations but no variation seems to help her circumvent breaking g, f or RSA. Yet, we cannot prove that rigorously!

Now, let us see how to pay Martin:

PROTOCOL 5.6.

1. Alice sends the coin signature C to Martin. $\xrightarrow{C}$
2. Martin chooses some random bit string $z \in \{0,1\}^{k/2}$ and sends it to Alice. $\xleftarrow{z}$
3. Alice computes her answer Z by revealing a half of each identity pair:

$$Z_i = \begin{cases} (a_i, c_i, y_i) & \text{if } z_i = 1, \\ (x_i, b_i, d_i) & \text{if } z_i = 0. \end{cases}$$

   She sends Z to Martin. $\xrightarrow{Z}$
4. Martin computes $x_i = g(a_i, c_i)$ or $y_i = g(b_i, d_i)$ according to the value of z_i. He then checks the signature $C^3 = \prod_{i \in \mathbb{N}_{<k/2}} f(x_i, y_i)$ according to (5.5). If everything is OK, he accepts the payment.

The protocol already guarantees that the equations (5.5) must be valid. Otherwise Martin does not accept the coin. Of course, Martin wants to deposit the coin at the bank which is simply done as follows:

PROTOCOL 5.7.

1. Martin sends the entire payment transcript (C, z, Z) to the bank.
2. The bank verifies that the coin is valid and then checks whether the coin has already been deposited by searching for a coin with the same C in her database. If she does not find the coin she puts 100 € on Martin's account and sends him a receipt.
   If, however, she finds a coin (C, z', Z') she detects a double spending. There are two cases:
   - If $z = z'$ and $Z = Z'$ then Martin tries to redeposit an already deposited coin.
   - If $z \neq z'$ then also $Z \neq Z'$ and the bank knows a complete quadruple (a_i, b_i, c_i, d_i) and $a_i \oplus b_i = u||v'$ reveals Alice' identity. The bank calls the police.

   The case $z = z'$ and $Z \neq Z'$ is highly improbable. If transmission errors can be excluded this can only happen if Alice knows a collision for g.

There are various possible scenarios of trying to cheat. One possible problem is that Alice cooperates with Mallory, another merchant. She tells him to use Martin's challenge and then Mallory goes to the bank with the very same transcript as Martin. The bank knows that either Mallory or Martin is lying but she cannot tell which one. And also she has no way to catch Alice. But this is no new story to us: it can be prevented by using a pseudo-random challenge z that depends on the merchant's account and date and time of the transaction. Then Mallory cannot simply use the same challenge for he would be easily spotted as the misbehaving merchant and sued.

One further problem is that so far the bank can easily frame Alice for double spending. She can simply perform all of Alice' and Martin's actions. This means that the scheme cannot have any legal significance and thus no bank will use the system as it was presented. To prevent that Alice simply signs her identity in u. Instead of (5.3) Alice uses

$$a_i \oplus c_i = u||\operatorname{sig}_A(u, s_i)||(v + i).$$

Note that Alice must use a new random value s_i for each signature to prevent the bank from simply copying her signed identity.

5.3. Brands (1999). The solution described in Brands (1999) is a system that uses a different supposedly hard problem as the basis of an e€ system. Brands has described several variants of this system with different focuses. Apart from a system with similar properties than the previous one there are also solutions that incorporate a smart card as an extended arm of the bank. The smart card has to be asked upon any payment and can prevent double spending a priori. But even if the smart card's secret is revealed to a malicious Alice she still has to face double spending detection as in the previous system.

The basis for Brands' system is a generalization of the so-called discrete logarithm problem. If you work in a group then exponentiation is easy to calculate. However, finding an exponent e satisfying $x^e = y$ in the group may be difficult. An example for groups with supposedly difficult discrete logarithm problem are the subgroups of $\mathbb{Z}_p^\times$ generated by an element of order q where both p and q are prime and large enough. For example, taking p as a 1024-bit prime and q as a 160-bit prime was considered to be safe a few years ago. Other groups with difficult discrete logarithm problem are elliptic curves of appropriate size. (The number of bits for a point should be 160 to 240 bits.) The generalization used here is called the *representation problem*.

> Suppose you are given several generators $g_1, \ldots, g_r$ and some $x \in G$.
>
> Find $e_1, \ldots, e_r \in \mathbb{N}_{<\#G}$ with
>
> $$x = g_1^{e_1} g_2^{e_2} \ldots g_r^{e_r}.$$

In case $r = 1$ this is simply the discrete logarithm problem, so we only use $r \geq 2$ here. Clearly, if we can solve the discrete logarithm problem in G then we can solve this problem. The inverse is only partially true. So assuming that this problem is difficult is a little more than assuming that finding discrete logarithms is difficult.

If you want to know more about the details then read section 4 in Brands (1999). The basic reasonings about how to detect double spending or to prevent framing Alice or … are similar than the one before.

In view of the next system let us emphasize one more point. This system does not use cut-and-choose to give the bank the necessary conviction that the final coin has Alice' identity embedded. Instead a clever use of exponents and generators guarantees that.

5.4. Ferguson (1994b). This system is based on the difficulty of RSA and a discrete logarithm problem but it also uses some hash functions at sensitive places to (hopefully) increase the security. Further, polynomial secret sharing is used in order to decrease the coin size without loss of security. The important part here is Martin's challenge size, it must be large enough to prevent repetitions. The challenge size in Chaum *et al.* (1989) was $k/2$ bits, so the size of the coin grows linearly with the wanted challenge size. Here the challenge size depends only on the chosen group and is thus typically not much larger than with, say, $k = 4$. But let us first explain the polynomial secret sharing and the system.

5.4.1. Polynomial secret sharing. Suppose there is some secret x that we want to give to a group of people. Yet, the secret is very valuable and we do not trust a single person far enough to give him the secret. Think of the access code of the central safe of a bank or the start code of nuclear weapons. The solution is to distribute the secret: each person only gets part of the secret. Now, we know that to determine a polynomial f of degree less than k over some field $\mathbb{F}$ we need to know k pairs $(x, f(x))$. By interpolation we can then recover f, in particular, say, $f(0)$. If we give one point $(x, f(x))$, $x \neq 0$, to each person then at least k of them must come together to recover the secret $f(0)$ and thus to be able to open the safe or to start the missile. Figure 5.1 shows a picture of a line over $\mathbb{F}_{257}$. Any two points determine the secret. But if we only know one point then any secret could complete the picture. In Figure 5.2 we see a line over $\mathbb{F}_{256}$, the elements of $\mathbb{F}_{256}$ have been numbered in some systematical way for that purpose. Again any two points determine the line, one point could

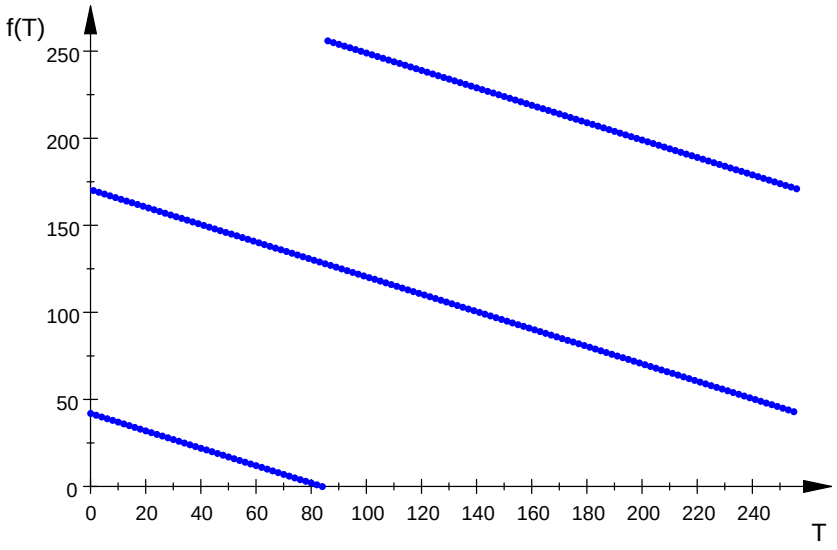


Figure 5.1: The line $f\colon \mathbb{F}_{257} \to \mathbb{F}_{257},\ T \mapsto 128\,T + 42$ over the field $\mathbb{F}_{257}$ carries the secret $f(0) \mathrel{\widehat{=}} 42$ and passes through zero at $T \mathrel{\widehat{=}} 84$. The elements of $\mathbb{F}_{257}$ are represented as integers modulo 257 (which is prime!).

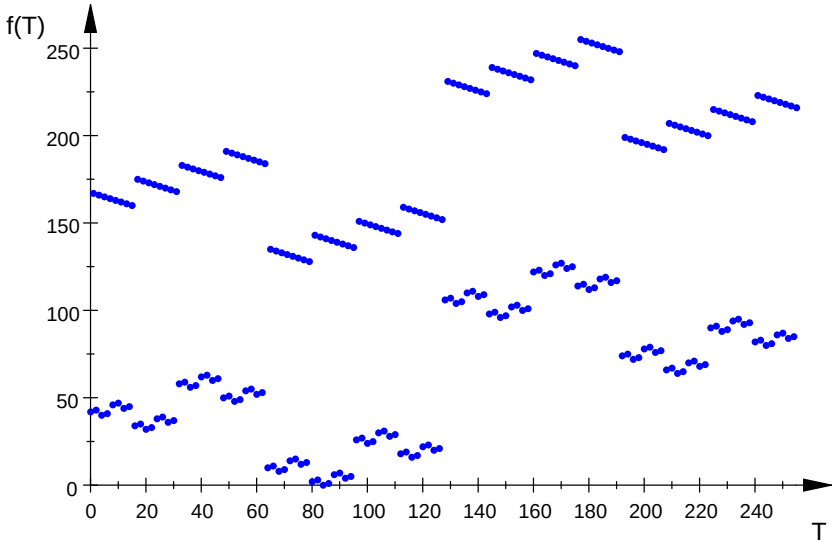


Figure 5.2: The line $f\colon \mathbb{F}_{256} \to \mathbb{F}_{256},\ T \mapsto \left(x^7 + x^3 + x^2 + 1\right)\,T + \left(x^5 + x^3 + x\right)$ over the field $\mathbb{F}_{256}$ carries the secret $f(0) = x^5 + x^3 + x \mathrel{\widehat{=}} 2^5 + 2^3 + 2 = 42$ and passes through zero at $T \mathrel{\widehat{=}} 84$. The elements of $\mathbb{F}_{256}$ are represented as polynomials in x of degree less than 8 over $\mathbb{F}_2 = \mathbb{Z}_2$ modulo $x^8 + x^4 + x^3 + x + 1$ and identified with integers by 'evaluating' such a polynomial over the integers at $x = 2$.

go with any secret. Figure 5.3 shows cubic curves. Only if we know at least four of its non-zero points then we can recover the secret.

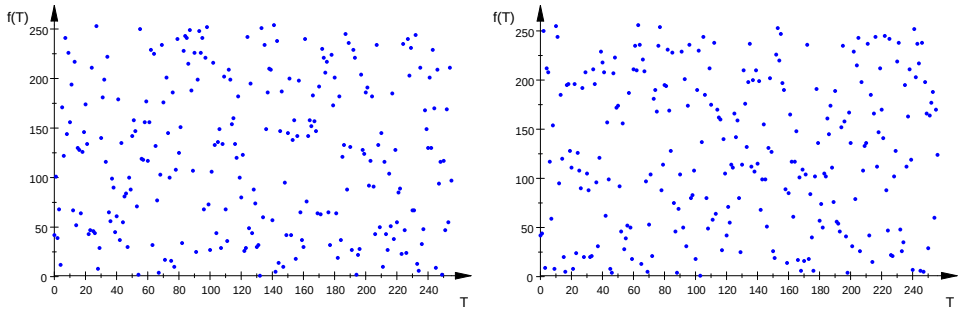


Figure 5.3: The cubic curve $f\colon \mathbb{F}_{256} \to \mathbb{F}_{256},\ T \mapsto (x^7+x^6+x^5+x^4+x^3+x+1)\,T^3 + (x^7+x^5+x+1)\,T^2 + (x^4+x^2+x+1)\,T + (x^5+x^3+x)$ over $\mathbb{F}_{256}$ on the left hand side and $f\colon \mathbb{F}_{257} \to \mathbb{F}_{257},\ T \mapsto 20\,T^3 + 42\,T^2 + (-60)\,T + 42$ over $\mathbb{F}_{257}$ on the right hand side each carry the secret $f(0) \mathrel{\widehat{=}} 42$. For our untrained eyes the nice structure of this curve is not visible but still: any four points determine the entire polynomial and thus the secret.

5.4.2. The system. Following the description of the author we also first describe the payment thus specifying the form of the coins. For the payment process we then have to find a way of getting the appropriate blind signatures from the bank. The basic setup contains an RSA signature key pair of the bank with public key (N, v). Additionally to the standard assumptions we require that v is a sufficiently large prime and that $\varphi(N)$ contains at least one large prime factor. Further some elements $g_1, g_2, g_3 \in \mathbb{Z}_N^\times$ of large order (minimal repetition length) are fixed. To be able to find them the bank should construct her primes p, q such that she knows large prime factors of $p-1$ and $q-1$. Next we need a suitable prime t such that $N \mid t-1$ and elements $h_2, h_3 \in \mathbb{F}_t^\times$ of order N. Finally, the bank chooses hash functions $f_1\colon \mathbb{Z}_N^\times \to \mathbb{N}_{<v}$, $f_2, f_3\colon \mathbb{F}_t^\times \to \mathbb{N}_{<v}$, and $f_4\colon \mathbb{N}_{<v} \times \mathbb{N}_{<v} \to \mathbb{Z}_N^\times$. The bank publishes the data

$$(N, v, g_1, g_2, g_3, t, h_2, h_3, f_1, f_2, f_3, f_4).$$

Further Alice' identity is coded in a value $U \in \mathbb{N}_{<v}$. Note that we will do a lot of calculations in the RSA domain $\mathbb{Z}_N^\times$ but some calculations also will take place in the field $\mathbb{F}_t$.

The coin consists of randomly chosen values $a, b, c \in \mathbb{Z}_N^\times$ from which anybody can compute $A = a g_1^{f_1(a)}$, $B = b g_2^{f_2(h_2^b)}$, $C = c g_3^{f_3(h_3^c)}$. Further a random parameter $k \in \mathbb{N}_{<v}$ and signatures $S_1 = (AC^k)^{(1/v)}$ and $S_2 = (BC^U)^{(1/v)}$ are part of the coin.

PROTOCOL 5.8. Payment.

1. Alice hands over (a, b, c) to Martin. $\xrightarrow{(a,b,c)}$
2. Martin chooses a random challenge $x \in \mathbb{N}_{<v}$. $\xleftarrow{x}$
3. Alice computes $r + \widehat{r}v \leftarrow kx + U$ with $r \in \mathbb{N}_{<v}$ and a signature R to $A^x BC^r$ by $R \leftarrow S_1^x S_2 C^{-\widehat{r}} = (A^x BC^r)^{(1/v)}$. She sends (r, R) to Martin. $\xrightarrow{(r,R)}$
4. Martin verifies that the signature is valid: all transmitted data are in the required domains and
$$R^v \stackrel{?}{=} A^x BC^r.$$
Note that he can do that.

Depositing the coin is easy, too:

PROTOCOL 5.9. Deposit.

1. Martin sends the entire transcript of the payment Protocol 5.8 to the bank.
2. She then looks up the signature in her database.
   - If she does not find it, Martin gets his money put on his account and a receipt.
   - Otherwise, the bank detects a double spending just as in the other systems:
     - If the challenges x and x' are also equal then Martin has tried to redeposit a coin.
     - Otherwise the bank tries to reveal Alice' identity. For now the bank knows $r \equiv_v kx+U$ and $r' \equiv_v kx' + U$ modulo v which is just a linear system of equations for k and U. Now she can take Alice to court for double spending.

There are several points to be taken into account for the withdrawal process. Of course the first requirement is that the bank cannot link the withdrawal and the deposit of a coin (unless a double spending occurs). Further, it shall be guaranteed that the parameters a, b, c and k are chosen randomly. Both parties, in particular the bank in our case, have to be sure that these parameters are not 'made up'. To do so Alice and the bank each choose a part, say a' and a'' of these parameters and at the end they take the product $a = a'a''$. Only both must make their choice independently whereas we have no way of guaranteeing a parallel transmission of the respective shares. (Actually, this seems very similar to 'Coin flipping by phone', Blum 1982.) To achieve this, Alice first chooses a' and then transmits some information $\tilde{A}$ which binds her to this value of a'. Then the bank chooses a'' and sends it to Alice. Actually, in our case the product must only be known to Alice. To make sure that Alice continues as desired, Alice sends something which requires that she uses the bank's a'' in order to give her the desired meaningful signature. Or the bank's answer depends on the information $\tilde{A}$ that binds Alice. Then the answer is only useful to Alice if she sticks to her previously chosen value a'.

5.4.3. Randomized blind signatures. First we consider how to get a *randomized blind signature*. Randomized means that the bank will be sure that the used parameter was indeed chosen at random. Blind means, as usual, that the bank cannot link the final signature to the transcript of the signature protocol. And of course Alice should not be able to generate such a signature on her own (this makes it a signature). Thus this scheme will be well suited for our needs. Ferguson attributes it to Chaum (1992). Additionally we use a one-way hash function $f\colon \mathbb{Z}_N^\times \to \mathbb{N}_{<v}$.

PROTOCOL 5.10. Randomized blind signature.

1. Alice randomly chooses $a', \alpha \in \mathbb{Z}_N^\times$ and $\sigma \in \mathbb{N}_{<v}$. She computes $\widetilde{A} \leftarrow \alpha^v a' g^\sigma$ and sends that to the bank. — $\xrightarrow{\widetilde{A}}$
2. The bank randomly chooses $a'' \in \mathbb{Z}_N^\times$ and sends it to Alice. — $\xleftarrow{a''}$
3. Alice computes $a \leftarrow a'a'' \in \mathbb{Z}_N^\times$ and an adjusting exponent $e + \widehat{e}v \leftarrow f(a) - \sigma$ with $e \in \mathbb{N}_{<v}$ and sends e to the bank. — $\xrightarrow{e}$
4. The bank computes $\overline{A} \leftarrow \widetilde{A} \cdot a'' g^e$ and sends Alice a signature $\widetilde{S} \leftarrow \overline{A}^{(1/v)}$ of it. — $\xleftarrow{\widetilde{S}}$
5. Alice unblinds the signature to obtain $S \leftarrow \widetilde{S}\alpha^{-1} g^{\widehat{e}}$. Now she has a signature pair (a, S) satisfying

$$S^v \stackrel{?}{=} a g^{f(a)}. \tag{5.11}$$

Before we discuss attacks let us have a short glance at the correctness. There is one complication that we did not mention in advance. Actually, Alice must hand over $e \in \mathbb{N}_{<v}$ instead of $e + \widehat{e}v$ in order to keep her secrets protected. Unfortunately, it is not allowed to calculate modulo v (or any other number Alice knows of) in the exponent of g. She only knows that g has large order but she has no idea which one. Thus she will obtain a v-th root of $a g^{f(a)-\widehat{e}v}$ instead of a v-th root of $a g^{f(a)}$.

Luckily this is correctable since the deviation is a v-th power of a known value. Indeed, we have

$$\begin{aligned} S^v &= \widetilde{S}^v \alpha^{-v} g^{\widehat{e}v} \\ &= \overline{A} \alpha^{-v} g^{\widehat{e}v} \\ &= \widetilde{A} \cdot a'' g^e \alpha^{-v} g^{\widehat{e}v} \\ &= \alpha^v a' g^\sigma \cdot a'' \alpha^{-v} g^{f(a)-\sigma} \\ &= a g^{f(a)}. \end{aligned}$$

First, note the relations between the values in the transcript: Clearly, Step 4 in Protocol 5.10 implies

$$\widetilde{S}^v = \widetilde{A} \cdot a'' g^e. \tag{5.12}$$

Everything else in the transcript is independent, as we will see shortly. Indeed, even if Alice follows the protocol any combination of $\widetilde{A}$, a'' and e can occur: First choose any value for a, then solve $e + \widehat{e}v = f(a) - \sigma$ for $\sigma \in \mathbb{N}_{<v}$ and $\widehat{e}$, $a = a'a''$ for a'', and $\widetilde{A} = \alpha^v a' g^\sigma$ for α. (We do not care for efficiency here!) Thus (5.12) is the only relation. Each protocol transcript even occurs with the same probability. The only choice is the choice of a, all other solutions are unique. Thus in order to obtain a valid signature from the protocol Alice can choose $\widetilde{A}$ and e but must then go along with a'' and $\widetilde{S}$ as given by the bank. Though Alice can choose $\widetilde{A}$ as a v-th power of something she knows, her major problem is that she does not know the v-th root of a'' and thus cannot correct this factor to her needs without breaking RSA.

What if the bank tries to trace Alice? Can she get any information on the pair (a, S) that is Alice' signature at the end? No, she cannot. Indeed, each such pair occurs with the same probability from the view of the bank. The bank knows $\widetilde{A}$, a'', e and $\widetilde{S}$. Suppose Alice gets (a, S). Then there is exactly one choice for Alice that can have produced this outcome: $\sigma \in \mathbb{N}_{<v}$ and $\widehat{e}$ are uniquely determined by $e + \widehat{e}v = f(a) - \sigma$, α by $S = \widetilde{S}\alpha^{-1} g^{\widehat{e}}$, and a' by $a = a'a''$. The equation $\widetilde{A} = \alpha^v a' g^\sigma$ is implied by (5.12): $\widetilde{A} = \widetilde{S}^v \cdot (a'')^{-1} g^{-e} = \alpha^v S^v g^{\sigma - f(a)} / a'' = \alpha^v a g^{f(a)} g^{\sigma - f(a)} / a'' = \alpha^v a' g^\sigma$.

Let us see what happens if Alice tries to cheat. Clearly, she cannot solve (5.11) after fixing a unless she breaks the bank's signature which is assumed to be infeasible. But can she use the signature generation with a more or less prescribed a? As already stated only (5.12) binds the values of the transcript. Suppose she wants to get along with a prescribed a. What would she have to do in order to get a signature for it? To satisfy (5.12) she must solve $a g^{f(a)} = \widetilde{A} \cdot a'' g^e$ for e. She can choose $\widetilde{A}$ in a clever way, yet only before she knows a''. Writing $e + \widehat{e}v = f(a) - \sigma$ the equation $a = \widetilde{A} \cdot a'' g^{-\sigma - \widehat{e}v}$ must be solved for σ. Actually no matter how she has chosen $\widetilde{A}$ the task is to compute a discrete logarithm. But of course the parameters will be adjusted such that computing a discrete logarithm with base g is not feasible. By trying several σ at random she might get control of some bits of a but no more. Thus there seems at least to be no obvious way for Alice to cheat.

If Alice tries to use some more of the structure she might try to use some multiple of a power of (5.12) to obtain a valid signature on some expression $a g^{f(a)}$:

$$D\widetilde{S}^{Ev} = D\widetilde{A}^E (a'')^E \cdot g^{eE}$$

First note that D can only help if Alice knows a v-th root but that does not lead her far. To be helpful she might try to adjust this such that

$$\begin{aligned} (\widetilde{A} a'')^E &= a g^t, \\ t + eE &= f(a) \end{aligned}$$

with some $t \in \mathbb{N}_{<v}$. Alice can use the first equation only after she knows a'', when $\widetilde{A}$ is already fixed. So the obvious way to solve these equations is to choose E and t and determine a by the first equation. The control over a she can obtain this way depends on her ability of computing discrete logarithms with respect to g or $\widetilde{A}a''$. Finally, the second equation determines e. However, that means that A, E and t must be chosen before e is transmitted.

Note that computing a discrete logarithm with base $\widetilde{A}a''$ might be feasible! If the order of $\widetilde{A}a''$ is smooth and can be determined efficiently then we can compute discrete logarithms efficiently and thus find a 'good' E. So we choose a, compute E and e. The order of the group $\mathbb{Z}_N^\times$ however is unknown to Alice and infeasible to find (unless she breaks RSA). The bank could adjust a'' a little to avoid very low order elements. Yet, this affects the distribution of a'' and might not be desirable. Probably, it is true anyway that most elements of $\mathbb{Z}_N^\times$ are difficult discrete logarithm bases provided $\varphi(N)$ contains large prime factors.

A way to stop Alice from even trying the just described manipulation is to change the scheme a little. In the previous 'attack', it was essential that Alice can compute $(a'')^E$. If we replace a'' by $h^{a''}$ then Alice cannot simply compute the corresponding $h^{((a'')^E)}$ from $h^{a''}$. Since we compute $a = a'a''$ in $\mathbb{Z}_N$ the order of h must divide N. But we are not bound to the domains already in use and simply choose a prime t such that $h \in \mathbb{F}_t$ of order N exists, that is $t = \rho N + 1$ for some $\rho \in \mathbb{N}$. Once t is found any element $x \in \mathbb{F}_t^\times$ raised to the power $\frac{t-1}{N}$ gives an element $h = x^{\frac{t-1}{N}}$ of order 1, p, q, or N. The bank can easily exclude the first three cases by checking $h \neq 1$, $h^p \neq 1$ and $h^q \neq 1$. A drawback of this is that Alice cannot verify that. She is only able to check $h^N = 1$ and $h \neq 1$. But this is not really severe because it is in the bank's interest to have an element of highest possible order there. Of course we now have to modify the definition of the hash function, we need $f\colon \mathbb{F}_t^\times \to \mathbb{N}_{<v}$. In total we have the following

PROTOCOL 5.13. Randomized blind signature without exponential attack.

1. Alice randomly chooses $a', \alpha \in \mathbb{Z}_N^\star$ and $\sigma \in \mathbb{N}_{<v}$. She computes $\widetilde{A} \leftarrow \alpha^v a' g^\sigma$ and sends that to the bank. $\xrightarrow{\widetilde{A}}$
2. The bank randomly chooses $a'' \in \mathbb{Z}_N^\times$, computes $\widetilde{h} \leftarrow h^{a''}$ and sends it to Alice. $\xleftarrow{\widetilde{h}}$
3. Alice computes an adjusting exponent $e + \widehat{e}v \leftarrow f(\widetilde{h}^{a'}) - \sigma$ with $e \in \mathbb{N}_{<v}$ and sends it to the bank. $\xrightarrow{e}$
4. The bank computes $\overline{A} \leftarrow \widetilde{A} \cdot a'' g^e$ and sends Alice a signature $\widetilde{S} \leftarrow \overline{A}^{(1/v)}$ of it along with a''. $\xleftarrow{a'', \widetilde{S}}$
5. Alice calculates $a \leftarrow a'a''$ and unblinds the signature to obtain $S \leftarrow \widetilde{S}\alpha^{-1} g^{\widehat{e}}$. Now she has a signature pair (a, S) satisfying

$$S^v \stackrel{?}{=} a g^{f(h^a)}. \tag{5.14}$$

OPEN QUESTION 5.15. *Could Alice in either variant obtain more signatures than the number of times she executes the protocol?*

5.4.4. Withdrawal. For the withdrawal process we will use the previous signature scheme three times in parallel. Actually, for signing a we use the simple version and for signing b and c we use the one which is protected against the exponential attack. The first will be protected by an additional factor derived from the other two. As in the above protocols, Alice and the bank will each choose a share of the three values. Yet, a will be furthermore linked to the other two. This procedure would hand over three signatures to Alice. As we already saw in Protocol 5.8 which defined the payment from Alice to Martin, Alice needs signatures of AB^k and AC^U. Since we are using the RSA scheme to compute signatures these two are merely combinations of the three signatures to A, B and C.

The bank must be sure that U is used as specified since this is the identity coded into the coin. It will enable the bank to trace Alice in case of a double spending. This will be guaranteed since the bank puts together the second signature as one for AC^U.

It is in Alice' interest that k is randomly chosen and only known to herself since this parameter protects her identity! If it were known to anyone else then after only one payment U could be computed. But also the bank shall be sure that this parameter is chosen at random because otherwise Alice could try to fit this parameter according to her needs. Thus the bank will only hand over a signature to $A^{1/k'} C^{k''}$ without any knowledge of k' but with almighty power over k''. Using $A^{1/k'}$ (implicitly) is made possible by choosing a as a k'-th power. Alice can later raise the result to the k'-th power and thus giving her a signature of $AC^{k'k''}$ as desired.

One problem arises again several times: Alice has to correct the exponents that shall be dealt with only modulo v. For example, this happens to $k'k''$. The final exponent to be used must be $k = (k'k'') \operatorname{rem} v$. Since the difference is a multiple of v in some exponent Alice can correct that even in the v-th root. As can be verified in the protocol the corrections $\widehat{e_2}$ and $\widehat{e_3}$ are either 0 or -1. But the corrections $\widehat{1}$, $\widehat{e_1}$, and $\widehat{k}$ use the entire range $\mathbb{N}_{<v}$.

PROTOCOL 5.16. Withdrawal.

1. Alice chooses random shares $a', b', c' \in_R \mathbb{Z}_N^\times$, random blinding bases $\alpha, \beta, \gamma \in_R Z_N^\times$, and random blinding exponents $\sigma, \tau, \varphi \in_R \mathbb{N}_{<v}$. She computes the blinded candidates $\widetilde{A} \leftarrow \alpha^v a' \cdot g_1^\sigma$, $\widetilde{B} \leftarrow \beta^v b' \cdot g_2^\tau$, $\widetilde{C} \leftarrow \gamma^v c' \cdot g_3^\varphi$ and sends them to the bank.
2. The bank chooses her random shares $a'', b'', c'' \in_R \mathbb{Z}_N^\times$ and sends a'', $\widetilde{h_2} \leftarrow h_2^{b''}$, $\widetilde{h_3} \leftarrow h_3^{c''}$ to Alice.
3. Alice computes

$$e_2 + \widehat{e_2} v \leftarrow f_2(\widetilde{h_2}^{b'}) - \tau \qquad \text{with } e_2 \in \mathbb{N}_{<v},$$
$$e_3 + \widehat{e_3} v \leftarrow f_3(\widetilde{h_3}^{c'}) - \varphi \qquad \text{with } e_3 \in \mathbb{N}_{<v}.$$

   and chooses $k' \in_R \mathbb{N}_{<v}^\times$. After computing $a \leftarrow (a'a'' \cdot f_4(e_2, e_3))^{k'}$ and $k^- \in \mathbb{N}_{<v}$, $\widehat{1} \in \mathbb{N}$ such that $k'k^- = 1 + \widehat{1}v$, she computes

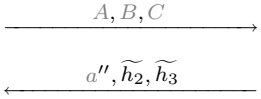
$$e_1 + \widehat{e_1} v \leftarrow k^- f_1(a) - \sigma \qquad \text{with } e_1 \in \mathbb{N}_{<v}.$$

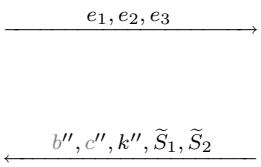
   Then she sends the exponents (e_1, e_2, e_3) to the bank.
4. The bank computes $\overline{A} \leftarrow \widetilde{A} a'' f_4(e_2, e_3) g_1^{e_1}$, $\overline{B} \leftarrow \widetilde{B} b'' g_2^{e_2}$, $\overline{C} \leftarrow \widetilde{C} c'' g_3^{e_3}$. Then the bank chooses her share $k'' \in_R \mathbb{N}_{<v}^\times$ of k. She then computes the signatures $\widetilde{S}_1 \leftarrow (\overline{A}\,\overline{C}^{k''})^{(1/v)}$ and $\widetilde{S}_2 \leftarrow (\overline{B}\,\overline{C}^{U})^{(1/v)}$ and sends them to Alice.
5. Alice puts everything together: she computes $b \leftarrow b'b''$, $c \leftarrow c'c''$ in $\mathbb{Z}_N^\times$, and $k + \widehat{k}v \leftarrow k'k''$ with $k \in \mathbb{N}_{<v}$. Now she can compute

$$A \leftarrow a g_1^{f_1(a)}, \quad B \leftarrow b g_2^{f_2(h_2^b)}, \quad C \leftarrow c g_3^{f_3(h_3^c)}$$

   and unblind the signatures $S_1 \leftarrow \left(\widetilde{S}_1 \left(\alpha^{-1} g_1^{\widehat{e_1}}\right) \left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^{k''}\right)^{k'} g_1^{-f_1(a)\widehat{1}} C^{-\widehat{k}}$ and $S_2 \leftarrow \widetilde{S}_2 \left(\beta^{-1} g_2^{\widehat{e_2}}\right) \left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^{U}$. She now has a coin (a, b, c, k, S_1, S_2) with the property

(5.17) $$S_1^v = A C^k, \quad S_2^v = B C^U.$$





First we verify that this indeed fulfills the claimed equations (5.17). What the bank obtains actually is

$$\overline{A}^{k'} = A \cdot \left(\left(\alpha g_1^{-\widehat{e_1}}\right)^{k'} g_1^{f_1(a)\widehat{1}}\right)^v,$$
$$\overline{B} = B \cdot (\beta g_2^{-\widehat{e_2}})^v,$$
$$\overline{C} = C \cdot (\gamma g_3^{-\widehat{e_3}})^v.$$

With this information we can exploit the definitions:

$$\begin{aligned} S_1^v &= \left(\left(\widetilde{S}_1 \left(\alpha^{-1} g_1^{\widehat{e_1}} \right) \left(\gamma^{-1} g_3^{\widehat{e_3}} \right)^{k''} \right)^{k'} g_1^{-f_1(a)\widehat{1}} C^{-\widehat{k}} \right)^v \\ &= \overline{A}^{k'} \left(\left(\alpha^{-1} g_1^{\widehat{e_1}} \right)^{k'} g_1^{-f_1(a)\widehat{1}} \right)^v \left(\overline{C} \left(\gamma^{-1} g_3^{\widehat{e_3}} \right)^v \right)^{k'k''} C^{-\widehat{k}v} \\ &= AC^{k'k''-\widehat{k}v} = AC^k \end{aligned}$$

and similarly

$$\begin{aligned} S_2^v &= \left(\widetilde{S}_2 \left(\beta^{-1} g_2^{\widehat{e_2}} \right) \left(\gamma^{-1} g_3^{\widehat{e_3}} \right)^U \right)^v \\ &= \overline{B} \left(\beta^{-1} g_2^{\widehat{e_2}} \right)^v \left(\overline{C} \left(\gamma^{-1} g_3^{\widehat{e_3}} \right)^v \right)^U \\ &= BC^U. \end{aligned}$$

Thus the key equations (5.17) hold.

In order to prevent the bank from framing an innocent Alice for double spending at some time Alice must provide a signature for this identity U. If this is not the case not only Alice will not trust the system but also the bank will not be able to prosecute Alice for a potential double spending. No court would blame Alice if she *can* be framed by the bank. Yet, we must somehow guarantee this in the withdrawal process, see below. The first thought how to implement this is to make U a signed version of Alice' identity. But then the bank cannot directly control that U has the correct form and thus Ferguson suggests a different approach.

5.4.5. Summary. Ferguson (1994b) uses polynomial secret sharing to allow many possible queries. To embed a polynomial $kx + U$ into the system we proceed like this: Three numbers a, b, c are chosen at random by the bank and Alice. For the mutual security it is important that each partner is sure that these figures are indeed random. This is done by something similar to 'coin flipping by phone'. Alice and the bank each choose a part of each number and the actual number then is composed of these two parts. Yet only Alice will know the outcome of the random number. This makes the system anonymous. From these numbers are derived three numbers A, B, C with the help of some one-way functions. This ensures that Alice has almost no influence on the specific values of these three numbers. In the withdrawal process the bank sends Alice RSA signatures for AC^k and BC^U. Here also k must be a random quantity and again both must be sure of it.

To answer a query x by Martin Alice shows a signature R to $(AC^k)^x(BC^U) = A^x BC^{kx+U}$. She can produce this new signature from the two she knows. Clearly, she must also hand over $r = kx+U$ since Martin cannot compute this quantity. If the bank gets two such answers the bank can solve for k and U and thus reveal Alice' identity coded in U.

That is a very brief sketch of the system. There are some complications in the way the numbers a, b, c and k are chosen and some technical details that are used to prevent certain kinds of attacks.

6. Further topics

Still there are completely different threats.

6.1. The perfect crime. Complete anonymity also has its drawbacks:

- Ed kidnaps a baby.
- He prepares 10 000 money orders for 1 000 € each and blinds them.

- ◦ He sends them to the authorities with the threat to kill the baby unless the following instructions are met: A bank signs all orders and the results are published in a newspaper (or by any other kind of broadcast).
- ◦ The authorities comply.
- ◦ Ed buys a newspaper, unblinds the orders and spends the coins. There is no way for the authorities to trace the money to him.
- ◦ Ed frees the baby.

6.2. Further properties. Fairness, re-usability, ...

6.3. 'Historical' remarks. DigiCash(David Chaum), eCash/Cybercash(Hettinga), flooz, ... To my knowledge no electronic cash system is used in practice.

6.4. Social aspects. Acceptance, necessity, environment.

6.5. Economical aspects. Transaction costs, existing concurrent paying systems.

7. Texts on electronic money

For the great congress several texts describing typically one system for electronic money have been considered. Finally, those marked with a circle '◦' have been used.

- – Chaum (1985) (surface description using lots of pictograms),
- ◦ Chaum, Fiat & Naor (1989) (unconditionally untraceable/anonymous electronic cash system [DigiCash]; no proofs),
- ◦ Ferguson (1993, 1994b) (nicely described protocols including some reasoning),
- – Ferguson (1994a) (extensions to Ferguson (1993, 1994b): multi-spendable coins, observers),
- – Brands (1993) (extensive paper containing Brands (1994a,b, 1995), section 11, 12 describe the basic system, sections 9, 10 are needed for details, sections 5, 6, 8 concern the underlying representation problem),
- – Brands (1994b) (with observers, complete protocols),
- – Brands (1994a) (with smart cards, coins and signatures are separated, uses Schnorr signatures, very small memory requirements),
- – Brands (1995) (off-line, no observer or smart card needed, uses Schnorr like signatures),
- ◦ Brands (1999), first four pages and section 4 (overview article, section 4 contains an example system similar (or equal?) to Brands (1995), section 2 describes preliminaries: modeling electronic cash, authentication techniques, [conventional dynamic authentication; dynamic authentication based on public key cryptography], section 3 dwells on electronic cash techniques: representing electronic cash, transferring electronic cash [transferring register based electronic cash; transferring electronic coins], when tamper-resistance is compromised [fraud detection; fraud tracing; fraud liability; fraud containment], security for account holders [preventing loss, preventing payment redirection, non-repudiation], privacy of payments [relaxed monitoring, anonymous accounts and anonymous devices; blinding; one-show blinding; guaranteeing your own privacy; one-sided versus two-sided untraceability]),
- – Medvinsky & Neuman (1993) (NetCash is a system to make various forms of electronic money be exchangeable and acceptable via various channels),
- – Frankel, Tsiounis & Yung (1996, 1998) (fair electronic cash),
- – Bellare, Garay, Jutla & Yung (1998a,b) (),

- Maitland & Boyd (2001) (use group signatures),
  - Schneier (1996), §6.4 (describes Chaum's system, advancing step by step)
- Kou (2003), §8.3 (describes Brands' system, probably Brands (1995)),
- Kou (2003), §8.4 (one-response digital cash),
  - Kou (2003), §8.5 (fair digital cash).

References

MANINDRA AGRAWAL, NEERAJ KAYAL & NITIN SAXENA (2003). PRIMES is in P. URL http://www.cse.iitk.ac.in/news/primality_v3.pdf. Preprint, v3.

M. BELLARE, J. GARAY, C. JUTLA & M. YUNG (1998a). VarietyCash: a Multi-purpose Electronic Payment System. In *Proceedings of the 3rd Usenix Workshop on Electronic Commerce*, BENNET S. YEE, editor, 17 pages. URL http://www.usenix.org/events/ec98/bellare.html. Full version is Bellare *et al.* (1998b).

M. BELLARE, J. GARAY, C. JUTLA & M. YUNG (1998b). VarietyCash: a Multi-purpose Electronic Payment System. URL http://www-cse.ucsd.edu/users/mihir/papers/vc.html. Proceedings version see Bellare *et al.* (1998a).

MANUEL BLUM (1982). Coin flipping by telephone. In *CRYPTO 1981*, 133–137. IEEE. ISSN 0000-0133. URL http://www-2.cs.cmu.edu/~mblum/research/pdf/coin/.

STEFAN A. BRANDS (1993). An Efficient Off-line Electronic Cash System Based On The Representation Problem. Technical report, Centrum voor Wiskunde en Informatica (CWI). URL http://citeseer.nj.nec.com/brands93efficient.html.

STEFAN BRANDS (1994a). Off-Line Cash Transfer by Smart Cards. Technical Report CS-R9455 1994, Centrum voor Wiskunde en Informatica. URL http://citeseer.nj.nec.com/brands94offline.html.

STEFAN BRANDS (1994b). Untraceable Off-Line Cash in Wallets with Observers. In *Advances in Cryptology: Proceedings of CRYPTO '93*, Santa Barbara CA, DOUGLAS R. STINSON, editor, number 773 in Lecture Notes in Computer Science. Springer-Verlag, New York. ISBN 0-387-57766-1. ISSN 0302-9743. URL http://link.springer.de/link/service/series/0558/bibs/0773/07730302.htm.

STEFAN BRANDS (1995). Off-Line Electronic Cash Based on Secret-Key Certificates. In *Proceedings of LATIN '95*, Valparaíso, Chile. Valparasio, Chili. URL http://citeseer.nj.nec.com/brands95offline.html.

STEFAN BRANDS (1999). Electronic Cash. In *Handbook on Algorithms and Theory of Computation*, MIKHAIL J. ATALLAH, editor, chapter 44. CRC Press, Boca Raton. ISBN 0-8493-2649-4. URL http://citeseer.ist.psu.edu/brands98electronic.html.

DAVID CHAUM (1985). Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM* **28**(10), 1030–1044. ISSN 0001-0782. URL http://portal.acm.org/citation.cfm?id=4373&coll=portal&dl=ACM&CFID=41194678&CFTOKEN=68326228.

DAVID CHAUM (1992). Randomized blind signature. Personal communication with Niels Ferguson.

DAVID CHAUM, AMOS FIAT & MONI NAOR (1989). Untraceable Electronic Cash (Extended Abstract). In *Advances in Cryptology: Proceedings of CRYPTO '88*, Santa Barbara CA. URL http://citeseer.nj.nec.com/chaum89untraceable.html.

C. C. COCKS (1973). A note on 'non-secret encryption'. CESG Memo. URL http://www.cesg.gov.uk/site/publications/media/notense.pdf.

WHITFIELD DIFFIE & MARTIN E. HELLMAN (1976). New directions in cryptography. *IEEE Transactions on Information Theory* **IT-22**(6), 644–654.

J. H. ELLIS (1970). The possibility of secure non-secret digital encryption.

J. H. ELLIS (1987). The history of Non-Secret Encryption. Communications-Electronics Security Group, part of the United Kingdom Civil Service (CESG). URL http://www.cesg.gov.uk/site/publications/media/ellis.pdf.

NIELS FERGUSON (1993). Single Term Off-Line Coins. Technical Report CS-R9318, Centrum voor Wiskunde en Informatica, Amsterdam. URL http://www.macfergus.com/pub/CS-R9318.html.

NIELS FERGUSON (1994a). Extensions of Single Term Coins. In *Advances in Cryptology: Proceedings of CRYPTO '93*, Santa Barbara CA, DOUGLAS R. STINSON, editor, number 773 in Lecture Notes in Computer Science, 292–301. Springer-Verlag, New York. ISBN 0-387-57766-1. ISSN 0302-9743. URL http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=773&spage=292.

NIELS FERGUSON (1994b). Single Term Off-Line Coins. In *Advances in Cryptology: Proceedings of EUROCRYPT 1993*, Lofthus, Norway, TOR HELLESETH, editor, number 765 in Lecture Notes in Computer Science, 318–328. Springer-Verlag, Heidelberg. ISBN 3-540-57600-2. ISSN 0302-9743. URL http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=765&spage=318.

YAIR FRANKEL, YIANNIS TSIOUNIS & MOTI YUNG (1996). "Indirect Discourse Proofs": Achieving Efficient Fair Off-Line E-Cash. In *Advances in Cryptology - ASIACRYPT '96*, K. KIM & T. MATSUMOTO, editors, number 1163 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg. ISBN 3-540-61872-4. ISSN 0302-9743. Also at http://citeseer.nj.nec.com/frankel96indirect.html and http://www.ccs.neu.edu/home/yiannis/papers/folc.ps.Z.

YAIR FRANKEL, YIANNIS TSIOUNIS & MOTI YUNG (1998). Fair Off-Line e-Cash made easy. In *Advances in Cryptology - ASIACRYPT'98*, K. OHTA & D. PEI, editors, number 1514 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg. ISBN 3-540-65109-8. ISSN 0302-9743. Also available at http://citeseer.nj.nec.com/frankel98fair.html and http://www.ccs.neu.edu/home/yiannis/papers/folc-es.ps.

JOACHIM VON ZUR GATHEN & JÜRGEN GERHARD (2003). *Modern Computer Algebra*. Cambridge University Press, Cambridge, UK, 2nd edition. ISBN 0-521-82646-2, 800. URL http://cosec.bit.uni-bonn.de/science/mca.html. First edition 1999.

WEIDONG KOU (editor) (2003). *Payment Technologies for E-Commerce*. Springer-Verlag, Berlin, Heidelberg, New York. ISBN 3-540-44007-0, X, 334 pp. URL http://www.springeronline.com/3-540-44007-0.

GREG MAITLAND & COLIN BOYD (2001). Fair Electronic Cash Based on a Group Signature Scheme. In *ICICS 2001*. URL http://sky.fit.qut.edu.au/~boydc/papers/GOC-ECash.pdf.

GENNADY MEDVINSKY & B. CLIFFORD NEUMAN (1993). NetCash: A design for practical electronic currency on the Internet. In *CCS'93: Proceedings of the First ACM Conference on Computer and Communications Security, November 1993, Fairfax, Virginia, United States, November 03-05, 1993.*, DOROTHY DENNING, RAY PYLE, RAVI GANESAN, RAVI SANDHU & VICTORIA ASHBY, editors, 102–106. ACM Press, New York. ISBN 0-89791-629-8. URL http://portal.acm.org/citation.cfm?id=168588.168601. See also http://citeseer.ist.psu.edu/medvinsky93netcash.html.

GENNADY MEDVINSKY & B. CLIFFORD NEUMAN (1994). Electronic Currency for the Internet. *ConneXions: the interoperability report* **8**(6), 19–23. ISSN 0894-5926. Also appeared as **?**.

R. L. RIVEST, A. SHAMIR & L. M. ADLEMAN (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM* **21**(2), 120–126.

BRUCE SCHNEIER (1996). *Applied cryptography : protocols, algorithms, and source code in C*. John Wiley & Sons, New York, 2nd edition. ISBN 0-471-12845-7, 0-471-11709-9, XXIII, 758.

A. SCHÖNHAGE & V. STRASSEN (1971). Schnelle Multiplikation großer Zahlen. *Computing* **7**, 281–292.

M. J. WILLIAMSON (1974). Non-secret encryption using a finite field. CESG Memo. URL http://www.cesg.gov.uk/site/publications/media/secenc.pdf.

M. J. WILLIAMSON (1976). Thoughts on cheaper non-secret encryption. CESG Memo. URL http://www.cesg.gov.uk/site/publications/media/cheapnse.pdf.

MICHAEL NÜSKEN
b-it (Bonn-Aachen International Center for Information Technology)
Dahlmannstr. 2
D53113 Bonn
Germany
nuesken@bit.uni-bonn.de
http://cosec.bit.uni-bonn.de/cosec/members/nuesken/