Lecture Notes

Esecurity: secure internet & electronic cash

Michael Nüsken

b-it

(Bonn-Aachen International Center for Information Technology)

Summer 2012
How are you? Norbert. Olivier
Your new resume will increase your salary several times. Confirm your real abilities and knowledge with the diploma. It is your chance.
Bachelor, Master, MBA, and Doctorate diplomas available in the field of your choice that's right, you can even become a doctor and receive all the benefits that come with it.
Many people share the same frustration, they are doing the work of the person that has the degree 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 degree of your dreams! You don't need to study or worry about passing those boring exams! You don't 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-code)

It is not necessary to detain progress of your life. Make a call and we will begin work. We work 24 hours a day for our clients.
Security?

- What do we have?
- What do we need?
- History and design?

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 email: 1971
- include address information about sender and receiver
- relay/forward email
- ease of use

→ DNS supplies information about the topology

→ Security?

→ SMTP Simple Mail Transfer Protocol
What do we need?

Security goals

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

DOS
- sender cannot deny the content: non-repudiation
- proof of delivery of submission of reading
- anonymity of the sender
- but still can't prove authentication

Our solution: GnuPG
**SAFETY**

- Reliability
  - random attacks
  - format syntax

**SECURITY**

- content semantics
- intentional attacks

---

**Attack (vs. Email)**

- Intercept emails
- Spoofing (pretend to be another sender)
- Phishing, social engineering
- Changing the message
- DNS spoofing
- Send malware

- **SPAM**
- **Denial of Service (DoS)**

---

**Defence**

- Encryption
- Signature

**CRYPTO**

- Canchon, knowledge
- Signature
- Encryption, Confirmation, support, DNSsec
- Antivirus software, sender trust + signature
- Filters, grep filtering,...

---

**EDUCATION**
Our toolbox: cryptography & related

1. Encryption

Alice (Austria)

Bob (Bonn)

Symmetric case: \( \text{key} = \text{key}' \)
Public-key case: \( \text{key}, \text{key}' \) are some two related
such that it is difficult to compute \( \text{key}' \) from \( \text{key} \)

Pros/Cons for public key:
- slower
- need no prior keys, can use insecure channels
- need much fewer keys.
Hybrid encryption unifies both and provides all pros!

Encryption protects against disclosure and prevents collusion.

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

Secrets are whatever remains unpredictable to an attacker.

Large scale internet competitions achieve \( \approx 2^{60} \) operations.
Signatures

Alice (Australia)

Bob (Boston)

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

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

Security: ?

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

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

Public key scenario: key1, key2 are different and it is difficult to derive key1 from key2.

Symmetric key scenario: key = key!
Major difference: in the symmetric scenario, the non-repudiation only holds partially. Only sender and receiver may generate signatures, but we cannot say who of the two did it unless we are one of sender or 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 symmetric scenario we call that message authentication code (MAC).
Public key infrastructures, web of trust, identity based mechanisms

How to trust a public key?

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

E.g. TLS/SSL used in HTTPS uses this kind of certification.

Idea:

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

Owner Address

Key Sign
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**
**Zertifikatshierarchie**

- Deutsche Telekom Root CA 2
- DFN-Verein PCA Global - G01
- Universität Bonn CA
cosec-srv-02.bit.uni-bonn.de

**Zertifikats-Layout**

- Erweiterter Schlüsselgebrauch
- Zertifikatsgegenstand-Schlüssel-ID
- Zertifizierungsstellen-Schlüsselidentifikator
- Zertifikatsgegenstand-Alternativ-Name
- CRL-Verteilungspunkte
- Zertifizierungsstellen-Informations-Zugriff
- Zertifikatsunterzeichnungs-Algorithmus

**Signaturwert des Zertifikats**

**Feld-Wert**

<table>
<thead>
<tr>
<th>Größe: 256 Bytes / 2048 Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>5d fc 4a ee 80 8f 4f 76 0f 38 33 a7 ce 59 c9 ad</td>
</tr>
<tr>
<td>5b 6d ac 20 b7 2d d5 90 aa 2f 9e 3a b3 48 8c 6a</td>
</tr>
<tr>
<td>4b fa 81 46 51 7a 37 a0 fd b6 dc 33 86 4f 8e 6f</td>
</tr>
<tr>
<td>90 64 6a 06 d1 76 95 1c c0 c4 6e 36 d8 10 4f 4b</td>
</tr>
<tr>
<td>72 3d 4b 23 80 83 2f ce 16 99 83 02 9c c3 ea 84</td>
</tr>
<tr>
<td>24 a9 d7 ab 55 76 fd 84 57 41 f5 e4 69 a2 c1 11</td>
</tr>
<tr>
<td>9f 3f 21 4e 18 e0 11 d9 31 ec 37 93 d8 c2 3d d6 a2</td>
</tr>
<tr>
<td>3b 9b 2b c7 7a 48 2c 3f 80 3c e6 86 46 c2 97</td>
</tr>
</tbody>
</table>

**Exportieren...**

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

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

This is what PGP and GnuPG use.

Additionally, here are key servers collecting & signed keys/certificates.

Option 3: Use identity based schemes, or so-called Certificateless schemes.
The security of a PKI lies in:

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

What to look at for a new cryptographic scheme?

- Correctness:
  the scheme fulfillfuls the wanted task if everything works as specified.
- Efficiency:
  all parts of your scheme should be suitably fast, i.e.
  polynomial-time from a theoretical perspective,
  within given time constraints, from a practical perspective.

(eg. 1s or 1min)
Security

A system is insecure if we can prevent 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 \(\epsilon\), i.e.,

\[
\Pr(\text{successful}) \geq \Pr(\text{guessing successful}) \geq \frac{1}{\text{poly}(k)}
\]

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

\[
\begin{align*}
\text{EX: Exploitation of a loop with each prob. } p. & \\
\text{EX? amplification}
\end{align*}
\]

(Actually, \(\Pr(\text{successful}) \leq \Pr(\text{guessing}) - \frac{1}{\text{poly}(k)}\))

may also be good!

We define the advantage of an attacker:

\[
\text{adv} = \left| \Pr(\text{successful}) - \Pr(\text{guessing}) \right|
\]
(3) We have to formulate what "successful" means.

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

Example: Public-key signatures.

Public info in particular the public keys of all users.

Chosen existential forgery

Security we require that no such attacker has polynomial advantage.
Example 1

RSA - signature:

\[ \text{Verify}(N,e)(m, \sigma) = (\sigma^e \equiv m) \text{ in } \mathbb{Z}_N \]

To make it a correct scheme we have to define

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

Commutative \( \checkmark \)

Efficiency \( \checkmark \)

Security ?

Is this scheme EUF-CMA secure?

No. It does this:

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

Example 2

RSA - FDH

Assume \( h : \{0,1\}^* \rightarrow \mathbb{Z}_N \)

is a cryptographic hash function

(collision resistant and one-way)

and (more or less) surjective.

Now let

\[ \text{Verify}(N,e)(m, \sigma) = (\sigma^e \equiv h(m) \text{ in } \mathbb{Z}_N). \]

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.
Theorem 1

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

Note: Security in the ROM $\not\rightarrow$ Security in the standard model.
Example: Public key encryption schemes

Decryption oracle

Ciphertext Attack

Public parameters and keys

Challenge oracle

Pick \( b \in \mathbb{Z}_p \)

\( c^* = \text{enc}(m_b^*) \)

Indistinguishability

\( \text{IND-CCA}_2 \)

\( d \in \mathbb{Z}_p \)

\( d = b \)

\( 1, c^* \neq 1, 4, \ldots, 3 \)

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

Good news: \( \text{NHA-CCA} \Leftrightarrow \text{IND-CCA} \)
At word on "efficient".

- asymptotic view
- fixed size view

versus

- trial division
- GNFS
- EC factoring

Factoring record:

768 bit, GNFS ➔ 2.5 years, parallel.

Award on hybrids:

EC hybrid encryption:
- RSA 2008 6.7
- AES 128 6.7

(Simple) combination has only 64 bit security.
Theorem

If $\mathcal{B}$ is a poly-time decider, then

- $\forall x \in \mathcal{B}$, $\mathcal{B}(x)$ is poly-time decidable.
- $\exists x \in \mathcal{B}$, $\mathcal{B}(x)$ is not decidable.

If $\mathcal{B}$ is decidable, then

- $\exists x \in \mathcal{B}$, $\mathcal{B}(x)$ is poly-time decidable.
- $\forall x \in \mathcal{B}$, $\mathcal{B}(x)$ is decidable.

Such an $\mathcal{B}$ is called a 'church decider'.
First note that it is easy to achieve a 50% success rate:

**GUesser**

- **Input:** \( N, e, y \)
- **Output:** \( b \in \{0, 1\} \)

1. \( b \in \{0, 1\} \)
2. Return \( b \).

We assume the \( B \) does less than this!

For simplicity, first assume \( B \) always finds the correct answer.

Let's see: how can we obtain \( B(T_e), \) say.

**Case** \( B(T_0)(x) = 0 \)

Then \( y = x^e \mod N \)

and \( x = 2x' \)

Of course, \( B(T_e)(x) = B(T_0)(x') \).

i.e. we have \( y = (2x')^e = 2^e \cdot x'^e \)

Thus \( y' = y \cdot 2^{-e} \mod N \)

is the ciphertext corresponding to \( x' \).

Calling \( B \) gives \( B(T_0)(x') = B(T_e)(x) \).

**Case** \( B(T_0)(x) = 1 \)

Write \( x = N - 2x' \)

thus \( y = (-2x')^e = (-2)^e \cdot x'^e \)

and ask \( B(T_e)(x) = B(T_e)(N) \oplus B(T_0)(x') \mod 2^e \)

so calling \( B \) with \( (N, e, y, (-2)^e) \)

reveals \( B(T_0)(x') \) and thus \( B(T_e)(x) \)
We can iterate this, or better call it recursively instead of B revealing the entire x instead of just its $B_{SD}$.

Difficulty: in general B is weaker than our assumption.

Solution: first amplify B such that its success probability $> 1 - c^k$.

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

The success rate of over all $k$ something like $(1 - c^k)^{\frac{1}{k}} \approx \exp(-1)$ exponentially close to 1.

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 $N_{SD}$ is necessary, where the attacker also just needs to find a single bit.
So let's check our options:

Is RSA IND-CCA$_2$ secure?

Is ElGamal?

**Theorem**

RSA is not IND-CCA$_2$ secure.

**Proof**

A $(k, r)$ picks $m_0^*, r^* = 1$, and the $c^* = 0$ if $b = 0$, $c^* = 1$ if $b = 1$.

So $(k, r)$ can answer correct in 50% of the cases.

**Proof**

$(k, r)$ picks $m_0^*$, $r^*$ uniformly, such that $m_0^* 
eq m^*$.

Then $(m_0^*)^e$ is the enc. of $m_0^*$

and it is known to $(k, r)$.

and $(m^*)^e$ is the enc. of $m^*$

and it is known to $(k, r)$.

So just compare $c^*$ to these two to get 100% correct answer.

**Proof**

Generalizes to any encryption scheme

where the encryption is deterministic,

i.e., to every plain text, there is at most one

ciphertext.

**Theorem**

No deterministic encryption scheme

can be IND-CCA$_2$ secure.

(Not even IND-CCA$_1$ secure.)
ElGamal encryption is not IND-CCA secure

Recall

\[ \text{enc: } P \text{ choose pair in a group } G, \]
\[ \text{inputs: } \text{A public key}, \]
\[ \text{M message} \]
\[ \text{output: } C = (T, D) \in G^2 \text{ cipher text} \]

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

Note: this is not deterministic!

This scheme has an additional feature:

Lemma: ElGamal encryption is homomorphic,

\[ \text{dec} (\text{enc } M_1 + \text{enc } M_2) = M_1 + M_2 \]

Given

\[ \text{enc } M_1 = (T_1, D_1) \]
\[ \text{where } T_1 = t_1 \cdot P, \ D_1 = M_1 + t_1 A \]
\[ \text{enc } M_2 = (T_2, D_2) \]
\[ \text{where } T_2 = t_2 \cdot P, \ D_2 = M_2 + t_2 A \]

Then

\[ \text{enc } M_1 + \text{enc } M_2 = (t_1 + t_2)P, \ M_1 + M_2 + (t_1 + t_2)A \]

And

\[ \text{dec} (\text{enc } M_1 + \text{enc } M_2) = [M_1 + M_2 + (t_1 + t_2)A - a \cdot (t_1 + t_2)P] \]
\[ = M_1 + M_2 \]

Thus one can use this to re-encrypt message by applying thelemma with \( M_2 = 0 \).
(Elliptic is not IND-CCA secure).

At picks $m^*_0, m^*_1$ at random, different.

The he asks for an encryption $c^*$ of one of the.

And now he re-encrypts $c^* : c_2 = c^* + 	ext{mac } 0$.

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 $m_2 = m^*_0$ asum 0,

if $m_2 = m^*_1$ asum 1.

celse: broken the challenge oracle
(can never happen!)

This attacks again is efficient and has 100% success.

Then Elliptic encryption actually

is IND-CCA secure,

provided the DLP of the used group is difficult.

To save the world: use a non-structured, reversible
operation when masking $M$!

For example, use

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

where $H$ is suitable hash function.

This yields (I think) an IND-CCA, secure

cryption scheme in ROF.
Summary

\[
\begin{align*}
\text{NPA} & 
\Downarrow 
\text{IND} 
\Downarrow 
\text{UBK} 
\Downarrow 
\text{KDA} 
\Downarrow 
\text{CCA}_2
\end{align*}
\]

in ROSE

or standard model

security notions for encrypt schemes.

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

Typical security proofs reductions do this:

Assume \( A \) is a successful attacker.

Construct a protocol \( R \) (reduction)

that may use \( A \) and solves some problem \( P \).
Secure connections

OSI

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

TCP/IP

- Transport
- Internet
- Network
- Host

- TCP, UDP, ...
- IPsec
- TLS/SSL
- SSH
- GnuPG

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

Pros/Cons of the placement in the layer model:

- The 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.
- Implementation: the lower, the more upper layers stuff may have to be redone, the easier/more difficult...
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 for the parameters,...

IPsec -> AH authentication header

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

IPsec -> ESP encapsulated security payload

- allows encryption
- allows authentication (no header)
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 deep packet inspection of higher layer info.
- NAT is problematic.
This is why we have AH and transport mode.

Many, many options.
IPv6 replaces IPv4 in order of Peer but Precipitated...
History of IKE

Protocol

NSA proposed ISAKMP

Validation Security Association Key Management

Not a protocol, just a framework

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

23 RFC

Difficult bread, notation changes, etc.
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

\[
\text{IP header} \quad \text{IP payload}
\]

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

\[
\begin{array}{c|c|c|c}
\text{new IP header} & \text{IPsec header} & \text{IP header} & \text{IP payload} \\
\end{array}
\]

In transport mode, only the IPsec header is added:

\[
\begin{array}{c|c|c}
\text{IP header} & \text{IPsec header} & \text{IP payload} \\
\end{array}
\]

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:

![Diagram of ESP header]

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 authentication 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

<table>
<thead>
<tr>
<th>IKE_SA initiator’s SPI</th>
<th>IKE_SA responder’s SPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next payload</td>
<td>Major version</td>
</tr>
<tr>
<td>Message ID</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Next payload</th>
<th>C</th>
<th>Reserved(0)</th>
<th>Payload length</th>
</tr>
</thead>
</table>

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.

<table>
<thead>
<tr>
<th>Exchange type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0-33</td>
</tr>
<tr>
<td>IKE_SA_INIT</td>
<td>34</td>
</tr>
<tr>
<td>IKE_AUTH</td>
<td>35</td>
</tr>
<tr>
<td>CREATE_CHILD_SA</td>
<td>36</td>
</tr>
<tr>
<td>INFORMATIONAL</td>
<td>37</td>
</tr>
<tr>
<td>Reserved to IANA</td>
<td>38-239</td>
</tr>
<tr>
<td>Reserved for private use</td>
<td>240-255</td>
</tr>
</tbody>
</table>
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 KEi = ga.

   Choose a nonce Ni.

2. Choose SAr1 from SAi1 unless no variant is supported.

<table>
<thead>
<tr>
<th>Next payload</th>
<th>Notation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>RESERVED</td>
<td></td>
<td>1-32</td>
</tr>
<tr>
<td>Security Association</td>
<td>SA</td>
<td>33</td>
</tr>
<tr>
<td>Key Exchange</td>
<td>KE</td>
<td>34</td>
</tr>
<tr>
<td>Identification - Initiator</td>
<td>IDi</td>
<td>35</td>
</tr>
<tr>
<td>Identification - Responder</td>
<td>IDR</td>
<td>36</td>
</tr>
<tr>
<td>Certificate</td>
<td>CERT</td>
<td>37</td>
</tr>
<tr>
<td>Certificate Request</td>
<td>CERTREQ</td>
<td>38</td>
</tr>
<tr>
<td>Authentication</td>
<td>AUTH</td>
<td>39</td>
</tr>
<tr>
<td>Nonce</td>
<td>Ni, Nr</td>
<td>40</td>
</tr>
<tr>
<td>Notify</td>
<td>N</td>
<td>41</td>
</tr>
<tr>
<td>Delete</td>
<td>D</td>
<td>42</td>
</tr>
<tr>
<td>Vendor ID</td>
<td>V</td>
<td>43</td>
</tr>
<tr>
<td>Traffic Selector - Initiator</td>
<td>TSi</td>
<td>44</td>
</tr>
<tr>
<td>Traffic Selector - Responder</td>
<td>TSr</td>
<td>45</td>
</tr>
<tr>
<td>Encrypted</td>
<td>E</td>
<td>46</td>
</tr>
<tr>
<td>Configuration</td>
<td>CP</td>
<td>47</td>
</tr>
<tr>
<td>Extensible Authentic</td>
<td>EAP</td>
<td>48</td>
</tr>
<tr>
<td>Reserved to IANA</td>
<td></td>
<td>49-127</td>
</tr>
<tr>
<td>Private use</td>
<td></td>
<td>128-255</td>
</tr>
</tbody>
</table>
Compute $\text{KE}_r = g^b$ if the group was guessed correctly. (Otherwise send:

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

)

Choose a nonce $\text{Nr}$.

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.

$$\text{SKEYSEED} = prf(\text{Ni} | \text{Nr} | g^{ab}),$$

$$\text{SK\_d|SK\_ai|SK\_ar|SK\_ei|SK\_er|SK\_pi|SK\_pr} = prf+(\text{SKEYSEED}, \text{Ni} | \text{Nr} | \text{SPI}_i | \text{SPI}_r)$$

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

$\text{SK\_d}$ is used for the derivation of keys in a child SA. $\text{SK\_ai}$ and $\text{SK\_ei}$ are used for authenticating and encrypting messages sent by the initiator, $\text{SK\_ar}$ and $\text{SK\_er}$ for messages sent by the responder.

4. The initiator send its identity $\text{ID}_i$, optionally one or more certificates $\text{CERT}$, a certificate request $\text{CERTREQ}$ (possibly including a list of trusted CAs), and optionally the responders identity $\text{ID}_r$ (it may be that the responder serves multiple identities ‘behind’ it).

Further she computes an authentication $\text{AUTH}$ (using the key from the first $\text{CERT}$ payload) for the entire first message concatenated with the responder’s nonce $\text{Nr}$ and the value $prf(\text{SK\_pi}, \text{ID}_i)$. 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 $\text{SA}_i^2$ with proposed traffic selectors $\text{TS}_i, \text{TS}_r$. 

<table>
<thead>
<tr>
<th>Next payload</th>
<th>Reserved(0)</th>
<th>Payload length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auth method</td>
<td>Reserved</td>
<td>Authentication data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hdr, SK</th>
</tr>
</thead>
</table>
| $\text{ID}_i, [\text{CERT},]$
| $[\text{CERTREQ},]$ |
| $[\text{ID}_r]$ |
| $\text{AUTH, SA}_i^2,$ |
| $\text{TS}_i, \text{TS}_r$ |
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.

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:

\[ \text{KEYMAT} = \text{prf}((\text{SK}_d, \text{Ni} | \text{Nr})) \]

### 2.2. Creating additional child SAs.

Further childs can be created under this IKE_SA using a CREATE_CHILD_SA exchange:

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:

\[ \text{Cookie} = \text{verID} | \text{hash}((\text{Ni}, \text{IPi}, \text{SPIi}, \text{secret}_{\text{verID}})) \]
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

<table>
<thead>
<tr>
<th>Next payload</th>
<th>Reserved(0)</th>
<th>Payload length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID type</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Identification data

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

<table>
<thead>
<tr>
<th>Next payload</th>
<th>Reserved(0)</th>
<th>Payload length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cert encoding</td>
<td>Certificate data</td>
<td></td>
</tr>
</tbody>
</table>

Certificate data
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.

![Diagram of main mode, public signature keys]

#### 3.2. Aggressive mode, public signature keys.

![Diagram of 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.

\[ K_A = \text{hash}(N_i, \text{cookie}_i) \]
\[ S_{Ai}, \{N_i\}_{Bob}, K_A\{KE_i\}, K_A\{ID_i\}, K_A\{CERT\} \]

\[ K_B = \text{hash}(N_r, \text{cookier}) \]
\[ S_{Ar}, \{N_r\}_{Alice}, K_B\{KE_r\}, K_B\{ID_r\}, \text{AUTH} \]

\[ SK = f(g^{ab}, N_i, N_r, \text{cookie}_i, \text{cookier}) \]
\[ SK\{AUTH\} \]

3.7. Main mode, pre-shared secret.

\[ S_{Ai} \]
\[ \leftarrow \]
\[ S_{Ar} \]
\[ \rightarrow \]
\[ K_{Ei}, N_i \]
\[ \leftarrow \]
\[ K_{Er}, N_r \]
\[ \rightarrow \]

\[ SK = f(\text{secret}, g^{ab}, N_i, N_r, \text{cookie}_i, \text{cookier}) \]
\[ SK\{ID_i, AUTH\} \]
\[ \leftarrow \]
\[ SK\{ID_r, AUTH\} \]

3.8. Aggressive mode, pre-shared secret.

\[ S_{Ai}, K_{Ei}, N_i, ID_i \]
\[ \leftarrow \]
\[ S_{Ar}, K_{Er}, N_r, ID_r, \text{AUTH} \]
\[ \rightarrow \]

\[ \text{AUTH} \]
\[ \leftarrow \]

\[ SK = f(\text{secret}, g^{ab}, N_i, N_r, \text{cookie}_i, \text{cookier}) \]

References


MICHAEL NÜSKEN
b-it, Bonn, Germany
Security questions

1. How secure?
   - By what definition? → Def, re.

2. Session key agreement
   - How long? Random? Unpredictable?
   - If the seed used for key generation is too short, the scheme is easily broken.
     (↑ Debian openSSL flaws)

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

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

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

- Do both parties contribute to it?
  - Psec: Yes.

Note: This is important to get strong security for each party.
Man in the middle?

P.S.: Not possible due to authentication.

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 anchor)

Basic idea:

Alice

Maurice

Rob

\[ A = xP \rightarrow A' \]

\[ B \leftarrow B = yP \]

\[ C = \text{sign}_{\text{Alice}}(A) \]

? \[ C_{A} \rightarrow \text{check that} \]

\[ Z_{B} \leftarrow \text{sign}_{\text{Bob}}(B) \]

? \[ Z_{B} \leftarrow \text{check that} \]

\[ Z_{B} = \text{sign}_{\text{Bob}}(B') \]

Maurice can break if he can solve the DH-problem.

... in which the signature scheme
Can an attacker decrypt recorded conversations given the long-term secrets after the termination of the session?

Psec: No, the attacker only finds the secret keys used for the authentication, but no remains of the short-term session keys which are entirely generated based on KE.

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

So here: no Forward Security.

To the contrary: Psec does provide Perfect Forward Security.

Escrow failure

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

GnuPG: Yes. No escrow failure here.

Psec: 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 failure.
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 - 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:

This may be exchanged periodically.
Then may put a server secret ID in the cookie

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

Impacts: It must be deterministic, suitable pseudorandom.
Endpoint identifies side 2.

1. 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.

2. Can an active attacker collect identity in formation?
   In principle, in one direction you cannot hide the identity. At best, you can hide one identity, viz. for a 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.
5. Live partner re-assurance

Is a replay attack possible?

Since nooses are used in the key generation, it is intractable to force the same key material. So we can assume live partners if we make nooses 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.

7. Stream protection

- How is the stream protected?  
  - Confidentiality: IPsec
  - Integrity
  - Authenticity

In IPsec provided we use: authentication, ESP, or AH.
Negotiating parameters?

IPsec: Yes!

Pros:
- flexible
- we exchange a cryptographic primitive with a better one. In particular, if one is broken.
- e.g. DES can be brute-force broken within a day by trying all 2^56 keys.

Cons:
- complexity
  - IPsec has a clear, single structure and hence will be analyzed easily enough.
  - downgrading by attackers
     - IPsec counters this by repeating the protocol again and again if you could use better protocol versions or algorithms.
- admins may choose badly

In summary, algorithms:

Keckhooffs 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".

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Bit Length</th>
<th>Status</th>
<th>Defined In</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1024 MODP Group</td>
<td>MUST-</td>
<td>[RFC2409]</td>
</tr>
<tr>
<td>14</td>
<td>2048 MODP Group</td>
<td>SHOULD+</td>
<td>[RFC3526]</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number</th>
<th>Defined In</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVED</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENCR_3DES</td>
<td>3</td>
<td>[RFC2451]</td>
<td>MUST-</td>
</tr>
<tr>
<td>ENCR_NULL</td>
<td>11</td>
<td>[RFC2410]</td>
<td>MAY</td>
</tr>
<tr>
<td>ENCR_AES_CBC</td>
<td>12</td>
<td>[AES-CBC]</td>
<td>SHOULD+</td>
</tr>
<tr>
<td>ENCR_AES_CTR</td>
<td>13</td>
<td>[AES-CTR]</td>
<td>SHOULD</td>
</tr>
</tbody>
</table>

3.1.4. IKEv2 Transform Type 2 Algorithms

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Number</th>
<th>Defined In</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVED</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRF_HMAC_MD5</td>
<td>1</td>
<td>[RFC2104]</td>
<td>MAY</td>
</tr>
<tr>
<td>PRF_HMAC_SHA1</td>
<td>2</td>
<td>[RFC2104]</td>
<td>MUST</td>
</tr>
<tr>
<td>PRF_AES128_CBC</td>
<td>4</td>
<td>[AESPRF]</td>
<td>SHOULD+</td>
</tr>
</tbody>
</table>
3.1.5. IKEv2 Transform Type 3 Algorithms

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Number</th>
<th>Defined In</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTH_HMAC_MDS_96</td>
<td>1</td>
<td>[RFC2403]</td>
<td>MAY</td>
</tr>
<tr>
<td>AUTH_HMAC_SHA1_96</td>
<td>2</td>
<td>[RFC2404]</td>
<td>MUST</td>
</tr>
<tr>
<td>AUTH_AES_XCBC_96</td>
<td>5</td>
<td>[AES-MAC]</td>
<td>SHOULD+</td>
</tr>
</tbody>
</table>

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


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.

<table>
<thead>
<tr>
<th>IPsec:</th>
<th>Encapsulating Security Payload (ESP) [RFC2406]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP encryption</td>
<td>TripleDES in CBC mode [RFC2451]</td>
</tr>
<tr>
<td>ESP integrity</td>
<td>HMAC-SHA1-96 [RFC2404]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IKE and IKEv2:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>TripleDES in CBC mode [RFC2451]</td>
</tr>
<tr>
<td>Pseudo-random function</td>
<td>HMAC-SHA1 [RFC2104]</td>
</tr>
<tr>
<td>Integrity</td>
<td>HMAC-SHA1-96 [RFC2404]</td>
</tr>
<tr>
<td>Diffie-Hellman group</td>
<td>1024-bit Modular Exponential (MODP) [RFC2409]</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>IPsec:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>ESP [RFC2406]</td>
</tr>
<tr>
<td>ESP encryption</td>
<td>AES with 128-bit keys in CBC mode [AES-CBC]</td>
</tr>
<tr>
<td>ESP integrity</td>
<td>AES-XCBC-MAC-96 [AES-XCBC-MAC]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IKE and IKEv2:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>AES with 128-bit keys in CBC mode [AES-CBC]</td>
</tr>
<tr>
<td>Pseudo-random function</td>
<td>AES-XCBC-PRP-128 [AES-XCBC-PRF-128]</td>
</tr>
<tr>
<td>Integrity</td>
<td>AES-XCBC-MAC-96 [AES-XCBC-MAC]</td>
</tr>
<tr>
<td>Diffie-Hellman group</td>
<td>2048-bit MODP [RFC3526]</td>
</tr>
</tbody>
</table>

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: Some blocks are encrypted with the same cipher text all the time. |

**CBC**

| Pro: Blocks are encrypted all the time |
| Pro: IV allows many encryptions of the same plain text |
| Con: If one cipher text is lost, it resynchronizes very fast, only one other block is jumbled. |
| Con: We must receive stuff in order. Otherwise we have buffer three faster blocks |
| Pro: There is a security proof conclusion |
**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:

- May reuse an encryption primitive.
A secure connection

key exchange

Diffie-Hellmann

need:
- "secure" group
- good (pseudo) random number generator
  (as source of truly random bits)
  unpredictable to an attacker

need:
- public key signatures
  (or shared key)
- PKI
  + social problems...

need:
- fast algorithms:
  - fast encryption:
    - e.g., AES
  - fast authentication:
    - integrity
    - e.g., HMAC, SHA1
    - AES-CBC-MAC
(a) Can we omit the key at the end?

No: EXTENSION attack!

hashed message

text  addition

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

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

text

do a collision for the unkeysed hash
function will also be a collision
for this 'MAC'.

Essentially, this points to the right

Security for a keyed hash function

cannot (or message authentication code)

No attacker shall be able to find
a collision - without knowing the key.

This is very similar to collision resistance
of (unkeysed) hash functions.

We shouldn't simplify the attacker's task by omitting
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 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 $2^{-80}$.

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

Key exchange oracle?

Signature oracle

Decryption oracle

Attacker

Setup

Public key info

Choose & encrypt oracle

either a forged signature
or a guess which prepared
msg was encrypted
or...

Difficulty: This is a complicated security assumption.
Example

Assume we have an secure (and-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:

\[
\begin{align*}
\text{ENC} & \quad \text{msg bits} \\
0 & \rightarrow 00 \\
1 & \rightarrow 10 \\
\text{randomly} & \\
\text{ENC}' & \\
\end{align*}
\]

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

If the first plaintext bit was 0 then this changes the intermediate plaintext 00 into 11 which does never occur in 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 does a who?

Horton's principle:
A signature or authentication value needs to depend on the meaning of the plain text.

How: How to combine the encryption and authentication?

Option:
- ATE: authentication, then encrypt.
- ETA: encrypt, then authentication.

1. ETA violates Horton's principle.

(a) 

\[
\begin{align*}
\text{plaintext} & \xrightarrow{\text{mac}} \text{mac} \\
\text{Enc} & \xrightarrow{\text{MAC}} \text{cipher text} \\
\text{Enc} & \xrightarrow{\text{MAC}} \text{cipher text} \\
\end{align*}
\]

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

(b) 

\[
\begin{align*}
\text{plaintext} & \xrightarrow{\text{mac}} \text{mac} \\
\text{Enc} & \xrightarrow{\text{MAC}} \text{cipher text} \\
\text{Enc} & \xrightarrow{\text{MAC}} \text{cipher text} \\
\end{align*}
\]

This is ok to Horton because ciphertext with MAC identifies the plaintext.
pro: The recipient saves the decryption work if the authentication fails.

But: Psec uses Ekd (2).

Bad news, as this seems to violate Horner's principle.

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

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

TL: S/SS:  
SSH:
Foms of money

- coins
- banknotes
- bank account
- checks
- stocks

- hardwax money
- software money

On the internet? E-commerce?
- usually account-based transactions (withdrawal, bank transfer)

Distinction
- traceable / untraceable
- online / offline
- instantaneous / with delay
- anonymous / non-anonymous
- transferable / not transferable
- anonymous + electronic?

difficult to replace?
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.

Solider about privacy vs. security

"Freedom is fragile. Handle it with care."

Physical cash identifiers

- Special printing
- Watermark
- Structured paper
- Serial number
- Silver strip
- Pictures made of drawn with very thin lines, many of them
- FOIL and body printing that perfectly together
- Characteristic colour
So how do we know what essential features are.

We want:

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

Basic scenario:

```
bank

     ┌─────────┐
     │         │
  →   │         │   ←
     │   user   │
     └─────────┘

The big problem is:

- bitstrings are
  1) easy to copy
  2) easy to compare/recognize

... blind signatures...
**TLS HANDSHAKE PROTOCOL**

**Client**

- **CLIENT HELLO**
  - Protocol version
  - Supported ciphers (List)
  - Supported compression (List)
  - Client random 28 bytes

**Server**

- **SERVER HELLO**
  - Protocol version
  - Selected cipher suite
  - Selected compression algorithm
  - Random data (28B)

**Certificate**
- Server key exchange
- Certificate request

**Server Hello done**

**Application data**

**Encrypted**

**Finished**
INFRASTRUCTURE OF TLS

TLS

HANDSHAKE PROTOCOL

TLS-CONNECTION
compression algo + param
encryption algo + param
MAC algo + param

TLS-record protocol

SECURITY CONSIDERATIONS

1. SESSION KEY AGREEMENT

2. PEER-TO-FORWARD SECURITY
   - RSA
   - DH

3. DENIAL OF SERVICE

4. MAN IN THE MIDDLE ATTACK

5. LIVE PARTNER ASSURANCE

6. ENDPOINT IDENTIFIER HIDING

7. STREAM PROTECTION

8. PLASISILE DEPENDIBILITY

9. 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 normally.
- 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.
Withdrawal protocol

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

On each order she adds a list of 75 pairs of identity bit strings, so that having a pair gives Alice' identity information.

Alice commits to each these 1466 bit strings (secret shares).

Alice blanks out the secret and hands the order 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 and blindly and gives it back to Alice.

Alice multiplies and so she has a valid coin.
Payment protocol

1. The sender signs the claim to the bank, the bank signs the claim.
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, Martin signs the bank.

Deposit protocol

1. Martin gives a complete transcript of the payment protocol to the bank.
2. The bank verifies all commitments:
   - its signature
   - the uniqueness string
   - the identity string, and have the openings to
     If the signature is wrong: refuse and call Police.
     If the uniqueness string is registered and the openings are identical than in the old claim, Martin tries to cheat.
The car now is composed of

- basic data: amount, serial t.
- many (eq. 73) tuples (xᵢ, yᵢ), i ∈ N₂₈

Rank signature on the previous
where actually those known values (aᵢ, bᵢ, cᵢ, dᵢ) such that

\[ xᵢ = g(aᵢ, cᵢ) \]
\[ yᵢ = g(bᵢ, dᵢ) \]

and

\[ C \text{ is a signature on } \prod_{i < 2₈} f(xᵢ, yᵢ) \]

and

\[ aᵢ \oplus bᵢ = v \parallel vᵢ \]

with Merkle identity 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.
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's 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 \(g\).

Michael Nüsken
b-it (Bonn-Aachen International Center for Information Technology)
Dahmannstr. 2
D53113 Bonn
Germany
nuesken@bit.uni-bonn.de
http://cosec.bit.uni-bonn.de/cosec/
members/nuesken/
Case: Alice tries to forge a coin, i.e., she tries to solve the equations

\[ C^3 = N \prod_{0 \leq k < \frac{b}{2}} f(x_i, y_i) \]

and

\[ x_i = g(a_i, c_i), y_i = g(b_i, d_i) \]

One of her options is to choose \( G(a_i, b_i, c_i, d_i) \) for \( i > 0 \) and the try to find \( (a_0, b_0, c_0, d_0) \) to fill the remaining operation:

\[ f(g(0_0, c_0), g(b_0, d_0)) = N \prod_{i > 0} f(x_i, y_i) \] (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

\[ f(g(a_0, c_0), g(b_0, d_0)) = f(g(a_1, c_1), g(b_1, d_1)) \]

with \( a_0 \oplus b_0 = \text{null} \oplus \text{null} \). Then she can use this to get a valid match which she might spend twice.
She chooses further values and withdraws a coin incorporating \( a_0, b_0, c_0, d_0 \) from the bank.

Then she spends this coin to some merchant partly with a challenge \( z \), say.

The she spends the coin with \( c_0, b_0, c_0, d_0 \) to a cheating merchant \( M_0 \) (maybe herself...).

Who picks the challenge \( z' = (z_0, z_1, z_2, \ldots) \).

Since that is a different challenge, the bank cannot accuse any merchant, but \( c_0 + b_0 \) or \( a_0 + b_0 \) does not reveal Alice's 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 pre-enchased if we assume that \( f \) and \( g \) are collision-resistant.
Actually, we assume that:

\[ f : \mathbb{Z}_N \times \mathbb{Z}_N \rightarrow \mathbb{Z}_N \]

e.g., one-way and collision-resistant.

And \( g \) shall be bijective on its first argument, i.e., it is fixed.

The first requirement assures that (neither) \( x \) nor \( y \) carry information about \( a \) or \( b \), respectively.

Actually, this also further implies that

\[ x_s = g ( a, c ) \]

is a perfectly binding and computationally binding commitment to \( a \)
(with randomness \( c \)).
for better protection we force the merchant to choose its bits as

\[ h(C, \text{ merchant's account}, \text{ date/time, some more bits}) \]

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

To get a coin where this half of committed pairs reveals a person's identity, however, Alice & Keefe cannot predict which bits will be different. Thus Alice has to send roughly half of the envelopes with a code: say the good/bad envelopes

\[ \frac{k-t}{k} \approx 2^{\frac{-t}{k}} \]

So if \( t \) is large enough the bank is well-guarded against this type of attack. We must assume \( t \approx k/4 \) so \( e^{-\frac{k}{2}} \approx e^{-\frac{1}{64}} \)
PROTOCOL. Withdrawal.

1. Alice chooses \( a_i, b_i, c_i, d_i, r_i \in \mathbb{Z}_N \) 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 \( x = \sum_{j \in \mathbb{N}_{<\log_2 N}} x_j 2^j \) mod \( N \) with \( x_j \in \{0, 1\} \) such that the integer \( \sum_{j \in \mathbb{N}_{<\log_2 N}} x_j 2^j \) is less than \( N \). Then use \( x_j \oplus y_j \) to define \( x \oplus y = \sum_{j \in \mathbb{N}_{<\log_2 N}} (x_j \oplus y_j) 2^j \). 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 \( v \cdot 2^{32} + (v + i) \) supposing that we need 32 bits for the counter. Alice computes

\[ x_i := g(a_i, c_i), \]
\[ y_i := g(b_i, d_i), \]
\[ B_i := r_i^3 f(x_i, y_i) \tag{0.2} \]

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

2. Now the bank chooses a random subset \( R \) of \( k/2 \) indices in \( \mathbb{N}_{<k} \) and sends \( R \) to Alice.

3. Alice opens the envelopes chosen by \( R \) by sending \( (a_i, b_i, c_i, d_i, r_i) \in \mathbb{R} \) to the bank.

4. The bank tests (0.1) and (0.2) for \( i \in R \). If this turns out well she computes

\[ s := \prod_{i \in R} B_i^{(1/3)} \]

and sends \( s \) to Alice. The bank charges Alice's account 100€ and increments the counter \( v \) by \( k \).

5. Alice can then easily unblind this signature and obtains \( C = s/\prod_{i \in R} r_i \).

She re-indexes the identity pairs \( (a_i, b_i, c_i, d_i) \) to the indices \( 0, \ldots, k/2 - 1 \) such that \( f(x_0, y_0) < f(x_1, y_1) < \cdots < f(x_{k/2-1}, y_{k/2-1}) \). Finally, Alice increments her copy of \( v \) by \( k \). Now she has a coin

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

which fulfills the condition that

\[ a_i \oplus b_i = u|\{V(i)\}, \]
\[ x_i = g(a_i, c_i), \]
\[ y_i = g(b_i, d_i), \]
\[ C^3 = \prod_{i \in \mathbb{N}_{<k/2}} f(x_i, y_i). \tag{0.3} \]
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 else 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!

\[ \rightarrow \text{Bad for the Bank!} \]

Need to ensure a proof that it was Alice who minted the coin.

To avoid this problem replace (C, i) by

\[ (\text{Sign}_i \text{Alice}) (\text{V}_{i+1}) \]
**Historical remarks**

Several cash systems have been tried to make real:

- Digi Cash (David Chaum)
- eCash/Cyber cash (Hettinga)
- flooz

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

Then, that a base is essentially based on the security of RSA.

Other scarce decent systems followed.

E.g.: Brands (1993) suggested various options all based on the

Represention Problem in group G

Typically: $G = \mathbb{Z}_q^*$

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

The problem is to find:

- given $h \in G < g_1, \ldots, g_r$ find a representation $h = g_{r_1} \cdots g_{r_k}$
This problem is closely related to the DLP.

Bendel actually many variants with additional properties

- with observers
- fair, transferable,
- "remote protection"

Complexity of Chaum, Fiat & Naor

- We work with RSA: \(k\) bit modulus \(N\).
- We need a certain of secret shares: \(k\).

Factoring RSA 703 in 2012 \(\times 1000\) core years.

Typical: \(k = 2048\), \(k = 256\).

\# bits communicated
- Alice - Bob: \(k \cdot \log_2 \frac{1}{2^k} + k\)
- Alice - Marku: \(\frac{k}{2} \cdot \log_2 \frac{1}{2^k} + k\)
- Marku - Bob: \(k\)

Size of can: public data: \(k\)
secret data: \(2k\)
Figure 1: The line \( f: \mathbb{F}_{257} \rightarrow \mathbb{F}_{257}, \; T \mapsto 128T + 42 \) over the field \( \mathbb{F}_{257} \) carries the secret \( f(0) \equiv 42 \) and passes through zero at \( T \equiv 84 \). The elements of \( \mathbb{F}_{257} \) are represented as integers modulo 257 (which is prime!).

Figure 2: The line \( f: \mathbb{F}_{256} \rightarrow \mathbb{F}_{256}, \; T \mapsto (x^7 + x^3 + x^2 + 1) \; T + (x^5 + x^3 + x) \) over the field \( \mathbb{F}_{256} \) carries the secret \( f(0) = x^5 + x^3 + x \equiv 2^5 + 2^3 + 2 = 42 \) and passes through zero at \( T \equiv 84 \). 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 \( x^3 + x^4 + x^3 + x + 1 \) and identified with integers by ‘evaluating’ such a polynomial over the integers at \( x = 2 \).
Figure 3: The cubic curve $f : \mathbb{F}_{257} \to \mathbb{F}_{257}$, $T \mapsto 20 T^3 + 42 T^2 + (-60) T + 42$ over $\mathbb{F}_{257}$ on the right hand side each carries the secret $f(0) \equiv 42$. 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 $f : \mathbb{F}_{256} \to \mathbb{F}_{256}$, $T \mapsto (x^7 + x^6 + x^5 + x^4 + x^3 + x + 1) T^3 + (x^7 + x^5 + x + 1) T^2 + (x^4 + x^2 + x + 1) T + (x^5 + x^3 + x)$ over $\mathbb{F}_{256}$ carries the secret $f(0) \equiv 42$. 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.
Requirements:

- Max's challenge must be large to be suitably unpredictable, manipulable:
  - Need exponentially many possible challenges

- Double-speed protection:
  - Need any two different challenges must reveal the secret.

- Each challenge must be answerable only with some information leaked by the hack using his secrets.

Idea:

Use secret sharing schemes that are more general than the five-bit scheme used in Chan, Fiat & Naor.
We consider for example for a line \( y = k \cdot x + U \) 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 matcher's challenge could be such as a value \( x \neq 0 \) and Alice's reply would be the corresponding \( y \).

Clearly: as long as Alice spends
the coin only once the pair \( (x, y) \) tells nothing about \( U \).
Because there is always a (unique) line through \( (x, y) \) and \( (0, U) \).

But if Alice spends the coin twice
then Bob obtains \( (x_1, y_1) \) and \( (x_2, y_2) \) both on this line.

Now the system
\[
\begin{align*}
\tau_1 &= k \cdot x_1 + U \\
\tau_2 &= k \cdot x_2 + U
\end{align*}
\]
demonstrates \( U \) (and \( k \)) if and only \( x_1 \neq x_2 \).
The core of Ferguson's solution uses three group elements $A, B, C,$ and two signatures of RSA type

\[ AC^k = S_1^r = \mathbb{Z}_N \]

\[ BC^u = S_2^r = \mathbb{Z}_N \]

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

Observe that

\[ (S_1 x S_2)^r = A C^k \cdot B C^u = A B C \]

\[ = A B C^r \]

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

This is not completely true if

\[ x \neq \phi(N)^k x + U \]
we need to make sure the signatures \( S \) and \( A^*B^C \) fit together without any break on Martin's side.

Clearly, if \( k \times u \in \mathbb{Z} \) then by the above we have

\[
(S_1^x S_2^y) = A^x B^y C^z.
\]

Now Alice writes \( k \times u = t + \frac{1}{n} \cdot v \),

the

\[
A^x B^y C^z = A^x B^y (k \times u) = (S_1^x S_2^y) \cdot C^z = (S_1^x S_2^y C^z)^v = S
\]

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 \( (1/v) = v^{-1} \mod L \).

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 \( x^{\frac{N}{g_3}} \) 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 \( t \equiv 1 \pmod{N} \). (Since the size of \( t \) needs only be of the same order as \( N \), she might simply choose the smallest prime of the form \( tN + 1 \).)

4. Find elements \( h_2, h_3 \in \mathbb{F}_t^\times \) of order \( N \). Again this is easy by testing the order of \( x^{\frac{N}{g_3}} \) for randomly chosen \( x \in \mathbb{F}_t^\times \) which can be only 1, \( p, q \), or \( N = pq \).

5. Fix hash functions \( f_1 : \mathbb{Z}_N^\times \to \mathbb{N}_{<v} \), \( f_2, f_3 : \mathbb{F}_t^\times \to \mathbb{N}_{<v} \), and \( f_4 : \mathbb{N}_{<v} \times \mathbb{N}_{<v} \to \mathbb{N}_{<v} \).

6. Publish \( \xi = (N, v, g_1, g_2, g_3, h_2, h_3, f_1, f_2, f_3, f_4) \).

The only extra information the bank needs is its secret exponent \( (1/v) \).

\[ \overline{\text{Some a way to compute:}} \]

\[ A = f_1(a) = a g_1 \]

\[ B = f_2(b) \]

\[ C = f_3(c) \]

based on the fixed hash functions.
PROTOCOL 2. Payment.

1. Alice hands over \((a, b, c)\) to Martin.

2. Martin chooses a random challenge \(x \in \mathbb{N}_{<v}\).

3. Alice computes \(r + \tilde{r}v \leftarrow kx + U\) with \(r \in \mathbb{N}_{<v}\) and a signature \(R\) to \(A^x BC^r\) by \(R \leftarrow S_1^x S_2 C^{-\tilde{r}} = (A^x BC^r)^{(1/v)}\). She sends \((r, R)\) to Martin.

4. Martin verifies that the signature is valid: all transmitted data are in the required domains and

\[R^v \equiv A^x BC^r.\]

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.
   - 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 redepone a coin.
     - Otherwise the bank tries to reveal Alice's 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.
How do we make blind signatures to a random quantum key?

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 \( A = C^g \).

Then from \( AC = S \),

she obtains \( A^{k + \theta} = S \),

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

\[ A^{r} B^\alpha C^\beta = R \]

Multiplying \( \bigcirc \) with \( \bigcirc \) we get

\[ A^{r} B^{\bar{\alpha}} C^{\bar{\beta}} = (R, S, C^{-g}) \]

where Alice may choose \( \alpha \) freely and adapts \( \beta \) such that \( r + \alpha (k + \theta) - \beta v \in N_v \).

So we must make sure that Alice does not have the power to choose A, B, C during the withdrawal to have an un-choosing.

But still we want that A, B, C are suitably random and un-known to the bank B.
You have studied or flipping by phone...
we want $\alpha \in \mathbb{G}_m^*$ to be chosen at random. This can be accomplished if three picks
$\alpha' \in \mathbb{G}_m^*$ and the bank picks $\alpha'' \in \mathbb{G}_m^*$
and the

$$\alpha = \alpha' \alpha''.$$ 

is uniformly random over $\mathbb{G}_m^*$
if only one of $\alpha'$ or $\alpha''$ is chosen uniformly random.

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

Additional problem: the bank must never know the out came!

Let's look at a solution: how to obtain a random $\alpha$ and a signature on $\frac{1}{\alpha} = g^\alpha$ (where $g^{2k} \neq 1$), from the bank setup such that $\alpha$ is unpredictable for the bank but she still be sure that her part $\alpha''$ has been used.
PROTOCOL 4. Randomized blind signature.

1. Alice randomly chooses $a', \alpha \in \mathbb{Z}_N^*$ and $\sigma \in \mathbb{N}_{<0}$. She computes $\widetilde{A} \leftarrow \alpha^\sigma a' g^\sigma$ and sends that to the bank. $\widetilde{A} \leftrightarrow a''$

2. The bank randomly chooses $a'' \in \mathbb{Z}_N^*$ and sends it to Alice. $\leftrightarrow e$

3. Alice computes $a \leftarrow a'a'' \in \mathbb{Z}_N^*$ and an adjusting exponent $e + \tilde{e}v \leftarrow f(a) - \sigma$ with $e \in \mathbb{N}_{<0}$ and sends $e$ to the bank. $\leftrightarrow \tilde{S}$

4. The bank computes $\overline{A} \leftarrow \widetilde{A} \cdot a'' g^e$ and sends Alice a signature $\overline{S} \leftarrow \overline{A}^{(1/\sigma)}$ of it.

5. Alice unblinds the signature to obtain $S \leftarrow \overline{S} \alpha^{-1} g^{-\tilde{e}v}$. Now she has a signature pair $(a, S)$ satisfying

\[ S^v \equiv a g^{f(a)}. \]
Relation in Protocol 4:

The honest bank makes sure that

\[ S^* = A^* a^n g^e \]

But this also follows from Alice's result

\[ S = a g \]

Actually, it is the only relation!

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

\[ A = a^* \]

\[ a'' = a = a'^a \]

\[ e = e + e' = f(a) - e \]

\[ S^* = S = A^* a^n g^e \rightarrow ok by (e) \]

seen from the bank's side: every \( a \) has the same probability. \( \Rightarrow \) Anonymity/Blinding

Since \( a = a' a'' \), the final outcome is uniformly random if only one party chooses its share uniformly.
PROTOCOL 5. Randomized blind signature without exponential attack.

1. Alice randomly chooses \( a', \alpha \in \mathbb{Z}_N^* \) and \( \sigma \in \mathbb{N}_{<v} \). She computes \( \bar{A} \leftarrow \alpha^a a' g^\sigma \) and sends that to the bank. \( \bar{A} \)

2. The bank randomly chooses \( a'' \in \mathbb{Z}_N^* \), computes \( \tilde{h} \leftarrow h^{a''} \) and sends it to Alice. \( \tilde{h} \)

3. Alice computes an adjusting exponent \( \varepsilon + \tilde{e} = f(\tilde{h}^a) - \sigma \) with \( e \in \mathbb{N}_{<v} \) and sends it to the bank. \( e \)

4. The bank computes \( \bar{A} \leftarrow \bar{A} \cdot a'' g^{e} \) and sends Alice a signature \( \bar{S} \leftarrow \bar{A}^{(1/\varepsilon)} \) of it along with \( a'' \). \( \bar{S}, a'' \)

5. Alice calculates \( a \leftarrow a' a'' \) and unblinds the signature to obtain \( S \leftarrow \bar{S} \alpha^{-1} g^{\tilde{e}} \). Now she has a signature pair \((a, S)\) satisfying

\[(**)
S^\varepsilon = ag^{f(\tilde{h}^a)}.
\]

1. Alice chooses random shares $a', b', c' \in_R \mathbb{Z}_N^n$, random blinding bases $\alpha, \beta, \gamma \in R \mathbb{Z}_N^n$, and random blinding exponents $\sigma, \tau, \varphi \in_R \mathbb{N}_{<v}$. She computes the blinded candidates $\vec{A} \leftarrow \alpha a' g_1^{x}, \vec{B} \leftarrow \beta b' g_2^{x}, \vec{C} \leftarrow \gamma c' g_3^{x}$ and sends them to the bank.

2. The bank chooses her random shares $a'', b'', c'' \in_R \mathbb{Z}_N^n$ and sends $a'', \vec{h}_2 \leftarrow h_2^{x''}, \vec{h}_3 \leftarrow h_3^{x''}$ to Alice.

3. Alice computes

$$e_2 + \varepsilon_2 x \leftarrow f_2(\vec{h}_2^{x''}) - \tau \quad \text{with } e_2 \in \mathbb{N}_{<v},$$

$$e_3 + \varepsilon_3 x \leftarrow f_3(\vec{h}_3^{x''}) - \varphi \quad \text{with } e_3 \in \mathbb{N}_{<v},$$

and chooses $k' \in_R \mathbb{N}_{<v}$. After computing $a \leftarrow (a'a'')^k$, $f_4(e_2, e_3)^{k'}$ and $k' \in \mathbb{N}_{<v}$, $\hat{t} \in \mathbb{N}$ such that $k'k' = 1 + \hat{t}v$, she computes

$$e_1 + \varepsilon_1 x \leftarrow k'f_1(a) - \sigma \quad \text{with } e_1 \in \mathbb{N}_{<v}.$$

Then she sends the exponents $(e_1, e_2, e_3)$ to the bank.

4. The bank computes $\vec{A} \leftarrow \vec{A}a'' f_3(e_2, e_3)g_1^{e_1}, \vec{B} \leftarrow \vec{B}b'' g_2^{e_2}, \vec{C} \leftarrow \vec{C}c'' g_3^{e_3}$. Then the bank chooses her share $k'' \in_R \mathbb{N}_{<v}$ of $k$. She then computes the signatures $\vec{s}_1 \leftarrow (\vec{A} \vec{C}^{k''})^{(1/v)}$ and $\vec{s}_2 \leftarrow (\vec{B} \vec{C}^{U})^{(1/v)}$ and sends them to Alice.

5. Alice puts everything together: she computes $b \leftarrow b'' b'', c \leftarrow c'' c' \in \mathbb{Z}_N^n$, and $k + k' \hat{t} v \leftarrow k'k''$ with $k \in \mathbb{N}_{<v}$. Now she can compute

$$A \leftarrow \alpha g_1^{f_1(a)}, \quad B \leftarrow bg_2^{f_2(\vec{h}_2^{x''})}, \quad C \leftarrow cg_3^{f_3(\vec{h}_3^{x''})}$$

and unblind the signatures $S_1 \leftarrow \vec{s}_1 \left(\alpha^{-1} \gamma^{x''} \right)^{k''} g_1^{-f_1(a)} \vec{C}^{-k}$ and $S_2 \leftarrow \vec{s}_2 \left(\beta^{-1} \gamma^{x''} \right)^{k''} g_2^{-f_2(\vec{h}_2^{x''})} \vec{C}^{U}$. She now has a coin $(a, b, c, k, S_1, S_2)$ with the property

$$(***) \quad S_1^v = AC^k, \quad S_2^v = BC^U.$$
\[ H' = (\tilde{A}^{\mu} \tilde{r}^e(\tilde{e}, \tilde{e}_5) \tilde{g}_{i}^{e_1}) e' \]

\[ \tilde{A} = \alpha^v g_{i} e^e \]

\[ \tilde{A} = \alpha^v g_{i} f_{i}(a) \]

\[ a = (\alpha^v f_{i}(a)) e' \]

\[ = a \left( \alpha^v g_{i} e^e \right) e' \]

\[ \varepsilon_i + \varepsilon_i v = k_{i} f_{i}(a) - \varepsilon_i \]

\[ = a \left( \alpha^v g_{i} e^e \right) e' \]

\[ k - k' = 1 + f_{i}(a) \]

\[ = a \left( \alpha g_{i} - \varepsilon_i v \right) \]

\[ A = A \left( \alpha g_{i} - \varepsilon_i v \right) \]

\[ B = B \left( \beta g_{i} - \varepsilon_i v \right) \]

\[ C = C \left( \gamma g_{i} - \varepsilon_i v \right) \]
So let's check $S_i^v = AC^k$:

$S_i^v = \left[ \left( S_i^v \left( \frac{\hat{a}}{g_i} \right) \left( g_i^{-1} \hat{a} \right)^{-k''} \right)^{k'} f_i(a)^{-v'} \right] C^{-v'}$

$\overset{\text{*}}{\sim} \overset{\text{v''}}{\sim} \sim S_i^v = AC^{k''} - k'^{-1} - k''^{-1} C^{-v'}$

$\overset{\text{v''}}{\sim} S_i^v = AC^{k''} - k'^{-1} - k''^{-1} C^{-v'}$

$\overset{\text{v''}}{\sim} S_i^v = AC^{k'' - k'^{-1} - k''^{-1} C^{-v'}}$

$= AC^{k'' - k' - k'' C^{-v'}}$

$= AC^{k'' - k' - k'' C^{-v'}}$

Similarly, we find that $S_2^v = BC^k$.

Bottom line: The Withdrawal is correct.

Efficiency: ok

Last question: Security?

1. How panic ty?

Same reasoning as with the original simple randomized blind signature scheme: assume a transcript is
given, choose a, b, c, d, and try to solve for the hidden parameters. Observe that there are exactly some number of solutions in each case.

2. Unforgeability?

We just note that all our tricks run into well-root extractions and/or disclose by arithmetic problems (or combinations).

We now have:

- probably un forgeable coins
- anonymous
- double spending protected
- very efficient and can size

Transaction protection for Alice has to be added. It cannot be simply embedded into C since the bank could stop 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’.


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:
   o its signature,
   o the uniqueness string,
   o 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 $f, g: \mathbb{Z}_N \times \mathbb{Z}_N \rightarrow \mathbb{Z}_N$ 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 \(a_i, b_i, c_i, d_i, r_i \in \mathbb{Z}_N\) for \(i \in \mathbb{N}_{<k}\) at random under condition that

\[
(5.3) \quad a_i \oplus b_i = u || (v + i).
\]

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 \(x = \left(\sum_{j \in \mathbb{N}_{<N}} x_j 2^j\right) \mod N\) with \(x_j \in \{0, 1\}\) such that the integer \(\sum_{j \in \mathbb{N}_{<N}} x_j 2^j\) is less than \(N\). Then use \(x_j \oplus y_j\) to define \(x \oplus y = \sum_{j \in \mathbb{N}_{<N}} (x_j \oplus y_j) 2^j\). 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

\[
\begin{align*}
  x_i &:= g(a_i, c_i), \\
  y_i &:= g(b_i, d_i), \\
  B_i &:= r_i^3 f(x_i, y_i)
\end{align*}
\]

for each \(i\) and sends the envelope vector \((B_i)\) to the bank.

2. Now the bank chooses a random subset \(R\) of \(k/2\) indices in \(\mathbb{N}_{<k}\) and sends \(R\) to Alice.

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.

4. The bank tests (5.3) and (5.4) for \(i \in R\). If this turns out well she computes

\[
s := \prod_{i \in R} B_i^{(1/3)}
\]

and sends \(s\) to Alice. The bank charges Alice’s account 100€ and increments the counter \(v\) by \(k\).

5. Alice can then easily unblind this signature and obtains \(C = s / \prod_{i \in R} r_i\). She re-indexes the identity pairs \((a_i, b_i, c_i, d_i)\) to the indices \(0, \ldots, k/2 - 1\) such that \(f(x_0, y_0) < f(x_1, y_1) < \cdots < f(x_{k/2 - 1}, y_{k/2 - 1})\). 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

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

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 \(\varepsilon\) of all her envelopes then the chance of not being caught is \(\left(\frac{k}{k/2}\right)^{k/2}\), which is at most \((1 - \varepsilon)^{k/2} \leq e^{-\frac{1}{2}\varepsilon k}\). On the other hand to hide her identity in double spending she must get a pair of challenges that differs only on that \(\varepsilon\) 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:

<table>
<thead>
<tr>
<th>(\varepsilon)</th>
<th>Chance of forgery</th>
<th>Chance for successful double spending</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>(2^{-64})</td>
</tr>
<tr>
<td>1/8</td>
<td>(2^{-17.54...})</td>
<td>(2^{-48})</td>
</tr>
<tr>
<td>1/4</td>
<td>(2^{-39.55...})</td>
<td>(2^{-32})</td>
</tr>
<tr>
<td>1/2</td>
<td>(2^{-124.17...})</td>
<td>1</td>
</tr>
</tbody>
</table>

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. 
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 (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 \( a_i \oplus b_i = u \| u' \) 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
\[
 a_i \oplus c_i = u \| \text{sig}_A(u, s_i) \| (v + i). 
\]
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

$$x = g_1^{e_1} g_2^{e_2} \cdots g_r^{e_r}.$$  

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

Figure 5.2: The line $f : \mathbb{F}_{256} \rightarrow \mathbb{F}_{256}, \ T \mapsto (x^7 + x^3 + x^2 + 1) T + (x^5 + x^3 + x)$ over the field $\mathbb{F}_{256}$ carries the secret $f(0) = x^5 + x^3 + x \equiv 2^5 + 2^3 + 2 = 42$ and passes through zero at $T \equiv 84$. 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 $f: \mathbb{F}_{256} \to \mathbb{F}_{256}$, $T \mapsto (x^7 + x^6 + x^5 + x^4 + x^3 + x + 1)T^3 + (x^7 + x^5 + x + 1)T^2 + (x^6 + x^2 + x + 1)T + (x^5 + x^3 + x)$ over $\mathbb{F}_{256}$ on the left hand side and $f: \mathbb{F}_{257} \to \mathbb{F}_{257}$, $T \mapsto 20T^3 + 42T^2 + (-60)T + 42$ over $\mathbb{F}_{257}$ on the right hand side each carry the secret $f(0) = 42$. 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 $g_1, g_2, g_3 \in \mathbb{Z}_N$ 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 $h_2, h_3 \in \mathbb{F}_t^\ast$ of order $N$. Finally, the bank chooses hash functions $f_1: \mathbb{Z}_N^\ast \to \mathbb{N}_{<v}$, $f_2, f_3: \mathbb{F}_t^\ast \to \mathbb{N}_{<v}$, and $f_4: \mathbb{N}_{<v} \times \mathbb{N}_{<v} \to \mathbb{Z}_N^\ast$. 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’s identity is coded in a value $U \in \mathbb{N}_{<v}$. Note that we will do a lot of calculations in the RSA domain $\mathbb{Z}_N^\ast$ but some calculations also will take place in the field $\mathbb{F}_t$.

The coin consists of randomly chosen values $a, b, c \in \mathbb{Z}_N^\ast$ from which anybody can compute $A = ag_1^{f_1(a)}$, $B = bg_2^{f_2(h_2)}$, $C = cg_3^{f_3(h_3)}$. Further a random parameter $k \in \mathbb{N}_{<v}$ and signatures $S_1 = (AC^k)^{(1/v)}$ and $S_2 = (BCU)^{(1/v)}$ are part of the coin.

**Protocol 5.8.** Payment.

1. Alice hands over $(a, b, c)$ to Martin.
2. Martin chooses a random challenge $x \in \mathbb{N}_{<v}$.
3. Alice computes $r + \tilde{r}v \leftarrow kx + U$ with $r \in \mathbb{N}_{<v}$ and a signature $R$ to $A^xBC^r$ by $R \leftarrow S_1^xS_2^rC^{-r} = (A^xBC^r)^{(1/v)}$. She sends $(r, R)$ to Martin.
4. Martin verifies that the signature is valid: all transmitted data are in the required domains and $R^v \equiv A^xBC^r$.

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 $r \equiv v \cdot kx + U$ and $r' \equiv vu \cdot kx' + U$ 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 $A$ 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 $A$ 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 $f : \mathbb{Z}_N^\times \rightarrow \mathbb{N}_{<v}$.

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

$$S^v \equiv ag^{f(a)^{1/v}} \equiv \hat{A}.$$  

(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 $e + \hat{c}v$ 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 $ag^{f(a)}\hat{c}v$ instead of a $v$-th root of $ag^{f(a)}$.
Luckily this is correctable since the deviation is a $v$-th power of a known value. Indeed, we have
\[
S^v = \tilde{S}^v \alpha^{- v} \tilde{\alpha}^v
= \bar{A} \cdot \alpha'' g^\sigma \alpha^{- v} \tilde{\alpha}^v
= \alpha'' \alpha' g^\sigma \cdot \alpha'' \alpha^{- v} g^{f(a) - \sigma}
= ag^{f(a)}.
\]

First, note the relations between the values in the transcript: Clearly, Step 4 in Protocol 5.10 implies
\[
(5.12) \quad \tilde{S}^v = \bar{A} \cdot a'' g^e.
\]

Everything else in the transcript is independent, as we will see shortly. Indeed, even if Alice follows the protocol any combination of $\tilde{A}$, $a''$ and $e$ can occur: First choose any value for $a$, then solve $\tilde{e} = f(a) - \sigma$ for $\sigma \in \mathbb{N}_{<v}$ and $\tilde{\alpha} = a' a''$ for $a''$, and $\bar{A} = \alpha' a'' g^\sigma$ for $\alpha$. (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 $\tilde{A}$, all other solutions are unique. Thus in order to obtain a valid signature from the protocol Alice can choose $\tilde{A}$ and $e$ but must then go along with $a''$ and $\tilde{S}$ as given by the bank. Though Alice can choose $\tilde{A}$ 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 $\tilde{A}$, $a''$, $e$ and $\tilde{S}$. Suppose Alice gets $(a, S)$. Then there is exactly one choice for Alice that can produce this outcome: $\sigma \in \mathbb{N}_{<v}$ and $\tilde{\alpha}$ are uniquely determined by $\tilde{e} + \sigma \tilde{\alpha} = f(a) - \sigma$, $\alpha$ by $S = S \tilde{\alpha}^{-1} g^\sigma$, and $a'$ by $a = a' a''$. The equation $\bar{A} = \alpha' a'' g^\sigma$ is implied by (5.12): $\bar{A} = \tilde{S}^v \cdot (a'')^{-1} g^{-e} = \alpha'' S^v g^{\sigma - f(a)/a''} = \alpha'' ag^{f(a)g^{n-f(a)/a''} = \alpha'' a' g^\sigma}$.

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 $ag^{f(a)} = \bar{A} \cdot a'' g^e$ for $e$. She can choose $\bar{A}$ in a clever way, yet only before she knows $a''$. Writing $\tilde{e} = f(a) - \sigma$ the equation $a = \bar{A} \cdot a'' g^{-\sigma - \tilde{\alpha}^v}$ must be solved for $\sigma$. Actually no matter how she has chosen $\bar{A}$ 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 $\sigma$ 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 $ag^{f(a)}$:
\[
D S^{E v} = D \bar{A}^E (a'')^E \cdot g^{E e}
\]
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
\[
(\bar{A} a'')^E = ag^t,
\]
\[
t + E e = f(a)
\]
with some $t \in \mathbb{N}_{<v}$. Alice can use the first equation only after she knows $a''$, when $\bar{A}$ 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 $\bar{A} a''$. 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 \( \tilde{a}a'' \) might be feasible! If the order of \( \tilde{a}a'' \) 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 \( h \) is not 1, \( h \) exists, and \( h = \rho N + 1 \) 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 \( h \neq 1 \).

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 \( A^{CU} \).

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.

---

**Protocol 5.13. Randomized blind signature without exponential attack.**

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

\[
S^{-1} \equiv ag^{f(h^{a''})}.
\]

---

**Open Question 5.15.** Could Alice in either variant obtain more signatures than the number of times she executes the protocol?
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 \( k = (k'k'') \text{rem } v \). 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 \( \hat{e}_2 \) and \( \hat{e}_3 \) are either 0 or -1. But the corrections \( \tilde{1}, \hat{e}_1, \) and \( \tilde{k} \) 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 \( \alpha, \beta, \gamma \in R \mathbb{Z}_N^\times \), and random blinding exponents \( \sigma, \tau, \varphi \in R \mathbb{N}_{<v} \). She computes the blinded candidates \( \tilde{A} \leftarrow \alpha a', \tilde{B} \leftarrow \beta b', \tilde{C} \leftarrow \gamma c' \) and sends them to the bank.

2. The bank chooses her random shares \( a'', b'', c'' \in R \mathbb{Z}_N^\times \) and sends \( a'', h_2 \leftarrow h_2'', \tilde{h}_3 \leftarrow h_3'' \) to Alice.

3. Alice computes
   \[
e_2 + \hat{e}_2 v \leftarrow f_2(h_2'') - \tau \quad \text{with } e_2 \in \mathbb{N}_{<v},
   \]
   \[
e_3 + \hat{e}_3 v \leftarrow f_3(h_3'') - \varphi \quad \text{with } e_3 \in \mathbb{N}_{<v}.
   \]
   and chooses \( k' \in R \mathbb{N}_{<v} \). After computing \( a \leftarrow (a' a'') \cdot f_4(e_2, e_3) \) and \( k'' \in \mathbb{N}_{<v} \), \( \hat{1} \in \mathbb{N} \) such that \( k'k'' = 1 + \hat{1} v \), she computes
   \[
e_1 + \hat{e}_1 v \leftarrow k'' f_1(a) - \sigma \quad \text{with } e_1 \in \mathbb{N}_{<v}.
   \]

Then she sends the exponents \( (e_1, e_2, e_3) \) to the bank.

4. The bank computes \( \tilde{A} \leftarrow \tilde{A} a'' f_4(e_2, e_3) g_1^{e_1}, \tilde{B} \leftarrow \tilde{B} b'' g_2^{e_2}, \tilde{C} \leftarrow \gamma c'' g_3^{e_3} \).

Then she sends the exponents \( h_2'' \leftarrow h_2'', \tilde{h}_3 \leftarrow h_3'' \) to Alice.

5. Alice puts everything together: she computes \( b \leftarrow b' b'', c \leftarrow c' c'' \in \mathbb{Z}_N^\times \),
   and \( \hat{k} v \leftarrow k'k'' \) with \( k \in \mathbb{N}_{<v} \). Now she can compute
   \[
   A \leftarrow a g_1^{f_1(a)}, \quad B \leftarrow b g_2^{f_2(h_2)}, \quad C \leftarrow c g_3^{f_3(h_3)}
   \]
   and unblind the signatures \( S_1 \leftarrow \left( S_1 \left( \alpha^{-1} g_1^{e_1} \right) \left( \gamma^{-1} g_3^{e_3} \right)^k \right)^k \)
   \[
   g_1^{f_1(a) \hat{1}} \left( g_1^{e_1} \right)^k \quad \text{and} \quad S_2 \leftarrow S_2 \left( \beta^{-1} g_2^{e_2} \left( \gamma^{-1} g_3^{e_3} \right)^U \right)^u \quad \text{where now she has a coin}
   (a, b, c, k, S_1, S_2) \) with the property
   \[
   (5.17) \quad S_1^u = AC^k, \quad S_2^u = BC^U.
   \]

First we verify that this indeed fulfills the claimed equations (5.17). What the bank obtains actually is
   \[
   \widetilde{A}^u = A \cdot \left( \alpha g_1^{e_1} k' f_1(a) \hat{1} \right)^v,
   \]
   \[
   \widetilde{B} = B \cdot \left( \beta g_2^{e_2} \right)^v,
   \]
   \[
   \widetilde{C} = C \cdot \left( \gamma g_3^{e_3} \right)^v,
   \]
With this information we can exploit the definitions:

\[ S_1^v = \left( \tilde{S}_1 \left( \alpha - g_4 \right) \left( \gamma - g_3 \right) k' \right)^v \]
\[ = A' \left( \left( \alpha - g_4 \right) \left( \gamma - g_3 \right) k' \right)^v \]
\[ = ACk'k'v = AC^k \]

and similarly

\[ S_2^v = \left( \tilde{S}_2 \left( \beta - g_2 \right) \left( \gamma - g_3 \right) U \right)^v \]
\[ = B' \left( \left( \beta - g_2 \right) \left( \gamma - g_3 \right) U \right)^v \]
\[ = BC^U. \]

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’s 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 \( AC^kx(BC^U) = AxBC^{kx+U} \). 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’s 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.


6.5. 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),
  − 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),
  − 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


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/