Security in Computer Networks

Multilateral Security in and by Distributed Systems

Transparencies for the Lecture:

Security and Cryptography I

(Version 2020/10/28)

Stefan Köpsell
(Slides mainly created by Andreas Pfitzmann)
Technische Universität Dresden, Faculty of Computer Science, D-01062 Dresden
Nöthnitzer Str. 46, Room 3062
Phone: +49 351 463-38272, e-mail: stefan.koepsell@dresden.de, https://dud.inf.tu-dresden.de/sac1
# Field of Specialization: Security and Privacy

<table>
<thead>
<tr>
<th>Lectures</th>
<th>Staff</th>
<th>SWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security and Cryptography I, II</td>
<td>Köpsell</td>
<td>2/2</td>
</tr>
<tr>
<td>Cryptography and -analysis</td>
<td>Franz</td>
<td>2/1</td>
</tr>
<tr>
<td>Information &amp; coding Theory</td>
<td>Franz</td>
<td>2/1</td>
</tr>
<tr>
<td>Data Security and Cryptography</td>
<td>Köpsell</td>
<td>0/4</td>
</tr>
<tr>
<td>Security Lab</td>
<td>Köpsell</td>
<td>2/2</td>
</tr>
<tr>
<td>Computers and Society</td>
<td>Köpsell</td>
<td>2/0</td>
</tr>
<tr>
<td>Seminar: Privacy and Security</td>
<td>Byrenheid et al.</td>
<td>2/0</td>
</tr>
<tr>
<td>Seminar: Secure app. development</td>
<td>Borcea-Pfitzmann</td>
<td>2</td>
</tr>
<tr>
<td>Seminar: Security in Computer Systems</td>
<td>Köpsell</td>
<td>2</td>
</tr>
<tr>
<td>Introduction to Data Protection Law</td>
<td>Wagner</td>
<td>2/0</td>
</tr>
</tbody>
</table>
Areas of Teaching and Research

- Multilateral security, in particular security by distributed systems
- **Privacy Enhancing Technologies (PETs)**
- Cryptography
- Physical Layer Security
- Information- and coding theory

- Security & Privacy
  - in Vehicular Networks (Connected Driving)
  - for IoT & Cyberphysical Systems
  - industrial communication
  - focused on humans: social engineering, transparency

- SDN & Cloud Security
Broadcast allows recipient anonymity — it is not detectable who is interested in which programme and information.
Examples of changes w.r.t. anonymity and privacy

Internet-Radio, IPTV, Video on Demand etc.
support profiling
Remark: Plain old letter post has shown its dangers, but nobody demands full traceability of them …
The massmedia „newspaper“ will be personalised by means of Web, elektronic paper and print on demand.
Areas of Teaching and Research

- Multilateral security, in particular security by distributed systems
- Privacy Enhancing Technologies (PETs)
- Cryptography
- Physical Layer Security
- Information- and coding theory

- Security & Privacy
  - in Vehicular Networks (Connected Driving)
  - for IoT & Cyberphysical Systems
  - industrial communication
  - focused on humans: social engineering, transparency

- SDN & Cloud Security
Aims of Teaching at Universities

Science shall clarify

*How something is.*

But additionally, and even more important

*Why it is such*

or

*How could it be*

(and sometimes, *how should it be*).

“*Eternal truths*” (i.e., knowledge of long-lasting relevance) should make up more than 90% of the teaching and learning effort at universities.
1. Education to **honesty** and a **realistic self-assessment**
2. Encouraging realistic **assessment of others**, e.g., other persons, companies, organizations
3. Ability to gather **security and data protection requirements**
   - Realistic protection goals
   - Realistic attacker models / trust models
Realistic protection goals/attacker models: Technical solution possible?
General Aims of Education in IT-security (sorted by priorities)

1. Education to **honesty** and a **realistic self-assessment**
2. Encouraging realistic **assessment of others**, e.g., other persons, companies, organizations
3. Ability to gather **security and data protection requirements**
   - Realistic protection goals
   - Realistic attacker models / trust models
4. **Validation** and **verification**, including their practical and theoretical **limits**
5. Security and data protection **mechanisms**
   - Know and understand as well as
   - Being able to develop

**In short:** *Honest IT security experts with their own opinion and personal strength.*
1. Education to **honesty** and a **realistic self-assessment**

   As teacher, you should make clear
   
   • your strengths and weaknesses as well as
   • your limits.

Oral examinations:

• Wrong answers are much worse than “I do not know”.
• Possibility to explicitly exclude some topics at the very start of the examination (if less than 25% of each course, no downgrading of the mark given).
• Offer to start with a favourite topic of the examined person.
• Examining into depth until knowledge ends – be it of the examiner or of the examined person.
1. Education to **honesty** and a **realistic self-assessment**
2. Encouraging realistic **assessment of others**, e.g., other persons, companies, organizations

Tell, discuss, and evaluate case examples and anecdotes taken from first hand experience.
1. Education to **honesty** and a **realistic self-assessment**
2. Encouraging realistic **assessment of others**, e.g., other persons, companies, organizations
3. Ability to gather **security and data protection requirements**
   - Realistic protection goals
   - Realistic attacker models / trust models

**Tell, discuss, and evaluate case examples (and anecdotes) taken from first hand experience.**

**Students should develop scenarios and discuss them with each other.**
1. Education to **honesty** and a **realistic self-assessment**
2. Encouraging realistic **assessment of others**, e.g., other persons, companies, organizations
3. Ability to gather **security and data protection requirements**
   - Realistic protection goals
   - Realistic attacker models / trust models
4. **Validation and verification**, including their practical and theoretical **limits**

Work on case examples and discuss them.

**Anecdotes!**
1. Education to **honesty** and a **realistic self-assessment**
2. Encouraging realistic **assessment of others**, e.g., other persons, companies, organizations
3. Ability to gather **security and data protection requirements**
   - Realistic protection goals
   - Realistic attacker models / trust models
4. **Validation** and **verification**, including their practical and theoretical **limits**
5. Security and data protection **mechanisms**
   - Know and understand as well as
   - Being able to develop

**Whatever students can discover by themselves in exercises should not be taught in lectures.**
...but no this way!

First stupid and silly
now wise as Goethe
this has accomplished
the power of the
Nuremberg Funnel

Nuremberg Funnel
(German: Nürnberger Trichter)
Postcard from around 1940
Offers by the Chair of Privacy and Data Security

- **Interactions** between **IT-systems** and **society**, e.g., conflicting legitimate interests of different actors, privacy problems, vulnerabilities ...
- Understand **fundamental security weaknesses** of today’s IT-systems
- Understand what **Multilateral security** means, how it can be characterized and achieved
- Deepened knowledge of the important tools to enable security in distributed systems based on **cryptography**
- Deepened knowledge in **error-free transmission and playback**
- Basic knowledge in **fault tolerance**
- Considerations in **building systems**: expenses vs. performance vs. security
- Basic knowledge in the relevant **legal regulations**
Aims of Education: Offers by other chairs

- Deepened knowledge **security in operating systems**
- **Verification** of OS kernels
- Deepened knowledge in **fault tolerance**
- Deepened knowledge in **trusted execution environments**
Table of Contents

1 Introduction
   1.1 What are computer networks (open distributed systems)?
   1.2 What does security mean?
      1.2.1 What has to be protected?
      1.2.2 Protection against whom?
      1.2.3 How can you provide for security?
      1.2.4 Protection measures – an overview
      1.2.5 Attacker model
   1.3 What does security in computer networks mean?

2 Security in single computers and its limits
   2.1 Physical security
      2.1.1 What can you expect – at best?
      2.1.2 Development of protection measures
      2.1.3 A negative example: Smart cards
      2.1.4 Reasonable assumptions on physical security
   2.2 Protecting isolated computers against unauthorized access and computer viruses
      2.2.1 Identification
      2.2.2 Admission control
      2.2.3 Access control
      2.2.4 Limitation of the threat “computer virus” to “transitive Trojan horse”
      2.2.5 Remaining problems

3 Cryptographic basics
Example: monitoring of patients, transmission of moving pictures during an operation

Why are legal provisions (for security and data protection) not enough?
History of Communication Networks (1)

1833 First electromagnetic telegraph
1858 First cable link between Europe and North America
1876 Phone operating across a 8.5 km long test track
1881 First regional switched phone network
1900 Beginning of wireless telegraphy
1906 Introduction of subscriber trunk dialing in Germany, realized by two-motion selector, i.e., the first fully automatic telephone exchange through electro-mechanics
1928 Introduction of a telephone service Germany-USA, via radio
1949 First working von-Neumann-computer
1956 First transatlantic telephone line
1960 First communications satellite
1967 The datex network of the German Post starts operation, i.e., the first communication network realized particularly for computer communication (computer network of the first type). The transmission was digital, the switching by computers (computer network of the second type).
1977 Introduction of the electronic dialing system (EWS) for telephone through the German Post, i.e., the first telephone switch implemented by computer (computer network of the second type), but still analogue transmission
1981 First personal computer (PC) of the computer family (IBM PC), which is widely used in private households

1982 Investments in phone network transmission systems are increasingly in digital technology

1985 Investments in telephone switches are increasingly in computer-controlled technology. Now transmission is no longer analogue, but digital signals are switched and transmitted (completed 1998 in Germany)

1988 Start-up of the ISDN (Integrated Services Digital Network)

1989 First pocket PC: Atari Portfolio; so the computer gets personal in the narrower sense and mobile

1993 Cellular phone networks are becoming a mass communication service

1994 www commercialization of the Internet

2000 WAP-capable mobiles for 77 € without mandatory subscription to services

2003 with IEEE 802.11b, WLAN (Wireless Local Area Network) and Bluetooth WPAN (Wireless Personal Area Network) find mass distribution

2004 UMTS starts in Germany

2005 VoIP (Voice over IP) is becoming a mass communication service

2007 first generation iPhone

2012 LTE with up to 300 MBit/s
Important Terms

**computers** interconnected by **communication network**

= **computer network** (of the first type)

**computers** providing switching in **communication network**

= **computer network** (of the second type)

**distributed** system

  spatial

  control and implementation structure

**open** system ≠ **public** system ≠ **open source** system

**service integrated** system

**digital** system
Development of the fixed communication networks of the German Post (Roadmap of approx. 1982)

**Services**
- Television
- View data
- TELEBOX
- Data transmission
- TELEFAX
- TEMEX
- Telex
- Teletex
- DATEX-L
- DATEX-P
- Videophone
- Video conference

**Networks**

**1986**
- Phone network
- Integrated text- and data network
- BIGFON

**Starting 1988**
- ISDN
- Video conference network

**Starting 1990**
- Broadband ISDN

**Starting 1992**
- Integrated broadband network

**Broadcast Networks**
- Communal aerial installations
- Broadband cable network
- Broadband cable network
- Switched networks
Threats and corresponding protection goals

**Threats:**

1) unauthorized access to information
   - Example: medical information system
   - Computer company receives medical files

2) unauthorized modification of information
   - Undetected change of medication
   - Integrity
   - \( \geq \) total correctness
   - \( \cong \) partial correctness

3) unauthorized withholding of information or resources
   - Detected failure of system
   - Availability for authorized users
   - No classification, but pragmatically useful
   - Example: unauthorized modification of a program

**Protection Goals:**

- Confidentiality
- Integrity
- Availability for authorized users

1) cannot be detected, but can be prevented; cannot be reversed
2) cannot be prevented, but can be detected; can be reversed
Threats and corresponding protection goals

**threats:**

1) unauthorized access to information
   - example: medical information system
   - computer company receives medical files

2) unauthorized modification of information
   - undetected change of medication

3) unauthorized withholding of information or resources
   - detected failure of system

**protection goals:**

- confidentiality
- integrity
- availability

- for authorized users
- ≥ total correctness
- ≅ partial correctness

no classification, but pragmatically useful

example: unauthorized modification of a program

1) cannot be detected, but can be prevented; cannot be reversed
2)+3) cannot be prevented, but can be detected; can be reversed
Definitions of the protection goals

**confidentiality**

Only **authorized users** get the **information**.

**integrity**

**Information** are **correct, complete, and current** or this is detectably not the case.

**availability**

**Information** and resources are accessible where and when the **authorized user** needs them.

- subsume: data, programs, hardware structure
- it has to be clear, who is authorized to do what in which situation
- it can only refer to the inside of a system
## Protection Goals: Sorting

<table>
<thead>
<tr>
<th>Content</th>
<th>Circumstances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prevent the unintended</td>
<td>Confidentiality</td>
</tr>
<tr>
<td></td>
<td>Hiding</td>
</tr>
<tr>
<td>Achieve the intended</td>
<td>Integrity</td>
</tr>
<tr>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td></td>
<td>Anonymity</td>
</tr>
<tr>
<td></td>
<td>Unobservability</td>
</tr>
<tr>
<td></td>
<td>Accountability</td>
</tr>
<tr>
<td></td>
<td>Reachability</td>
</tr>
<tr>
<td></td>
<td>Legal Enforceability</td>
</tr>
</tbody>
</table>
Observation of communication relations may give information about contents.

> Why encryption is not enough

Attorney Miller, specialized in mergers.
Protection Goals: Definitions

Confidentiality ensures that nobody apart from the communicants can discover the content of the communication.

Hiding ensures the confidentiality of the transfer of confidential user data. This means that nobody apart from the communicants can discover the existence of confidential communication.

Anonymity ensures that a user can use a resource or service without disclosing his/her identity. Not even the communicants can discover the identity of each other.

Unobservability ensures that a user can use a resource or service without others being able to observe that the resource or service is being used. Parties not involved in the communication can observe neither the sending nor the receiving of messages.

Unlinkability ensures that an attacker cannot sufficiently distinguish whether two or more items of interest (subjects, messages, actions, ...) are related or not.

Integrity ensures that modifications of communicated content (including the sender’s name, if one is provided) are detected by the recipient(s).

Accountability ensures that sender and recipients of information cannot successfully deny having sent or received the information. This means that communication takes place in a provable way.

Availability ensures that communicated messages are available when the user wants to use them.

Reachability ensures that a peer entity (user, machine, etc.) either can or cannot be contacted depending on user interests.

Legal enforceability ensures that a user can be held liable to fulfill his/her legal responsibilities within a reasonable period of time.
Additional Data Protection Goals: Definitions
(Rost/Pfitzmann 2009)

**Transparency** ensures that the data collection and data processing operations can be planned, reproduced, checked and evaluated with reasonable efforts.

**Intervenability** ensures that the user is able to exercise his or her entitled rights within a reasonable period of time.
Correlations between protection goals

Confidentiality
- Hiding
- Integrity
- Availability

Anonymity
- Unobservability

Accountability
- Reachability
- Legal Enforceability

implies
+ strengthens
- weakens
Correlations between protection goals

Confidentiality

Hiding

Anonymity

Unobservability

Integrity

Accountability

Reachability

Legal Enforceability

Availability

Transitive closure to be added
Transitive propagation of errors and attacks

A used B to design C

machine X executes program Y

transitive propagation of “errors”
universal Trojan horse

- commands
- unauthorized disclosure of information
- unauthorized modification of information
- unauthorized withholding of information or resources

(covert)
input channel

universal

(covert)
output channel
write access
non-termination
resource consumption
Protection against whom?

Laws and forces of nature
- components are growing old
- excess voltage (lightning, EMP)
- voltage loss
- flooding (storm tide, break of water pipe, heavy rain)
- change of temperature ...

Human beings
- outsider
- user of the system
- operator of the system
- service and maintenance
- producer of the system
- designer of the system
- producer of the tools to design and produce
- designer of the tools to design and produce
- producer of the tools to design and produce
  the tools to design and produce
- designer ...
  includes user, operator, service and maintenance ...

fault tolerance

Trojan horse
- universal
- transitive
<table>
<thead>
<tr>
<th>protection concerning protection against</th>
<th>to achieve the intended</th>
<th>to prevent the unintended</th>
</tr>
</thead>
<tbody>
<tr>
<td>designer and producer of the tools to design and produce</td>
<td>intermediate languages and intermediate results, which are analyzed independently</td>
<td>restrict physical access, restrict and log logical access</td>
</tr>
<tr>
<td>designer of the system</td>
<td>see above + several independent designers</td>
<td></td>
</tr>
<tr>
<td>producer of the system</td>
<td>independent analysis of the product</td>
<td></td>
</tr>
<tr>
<td>service and maintenance</td>
<td>control as if a new product, see above</td>
<td></td>
</tr>
<tr>
<td>operator of the system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>user of the system</td>
<td>physical and logical restriction of access</td>
<td></td>
</tr>
<tr>
<td>outsiders</td>
<td>protect the system physically and protect the data cryptographically from outsiders</td>
<td></td>
</tr>
</tbody>
</table>
### Which protection measures against which attacker?

<table>
<thead>
<tr>
<th>Protection concerning protection against</th>
<th>To achieve the intended</th>
<th>To prevent the unintended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer and producer of the tools to design and produce</td>
<td>intermediate languages and intermediate results, which are analyzed independently</td>
<td></td>
</tr>
<tr>
<td>Designer of the system</td>
<td>see above + several independent designers</td>
<td></td>
</tr>
<tr>
<td>Producer of the system</td>
<td>independent analysis of the product</td>
<td></td>
</tr>
<tr>
<td>Service and maintenance</td>
<td>control as if a new product, see above</td>
<td></td>
</tr>
<tr>
<td>Operator of the system</td>
<td>restrict physical access, restrict and log logical access</td>
<td></td>
</tr>
<tr>
<td>User of the system</td>
<td>physical and logical restriction of access</td>
<td></td>
</tr>
<tr>
<td>Outsiders</td>
<td>protect the system physically and protect data cryptographically from outsiders</td>
<td></td>
</tr>
</tbody>
</table>

**Physical distribution and redundancy:**

Confidentiality, unobservability, anonymity, unlinkability:

Avoid the ability to gather “unnecessary data”
It's not possible to protect against an omnipotent attacker.

- roles of the attacker (outsider, user, operator, service and maintenance, producer, designer …), also combined
- area of physical control of the attacker
- behavior of the attacker
  - passive / active
  - observing / modifying (with regard to the agreed rules)
- stupid / intelligent
  - computing capacity:
    - not restricted: computationally unrestricted
    - restricted: computationally restricted
Observing vs. modifying attacker

Observing attacker:
- Acting according to the agreed rules

Modifying attacker:
- Possibly breaking the agreed rules
Strength of the attacker (model)

Attacker (model) $A$ is stronger than attacker (model) $B$, iff $A$ is stronger than $B$ in at least one respect and not weaker in any other respect.

Stronger means:
- set of roles of $A \supset$ set of roles of $B$,
- area of physical control of $A \supset$ area of physical control of $B$,
- behavior of the attacker
  - active is stronger than passive
  - modifying is stronger than observing
- intelligent is stronger than stupid
  - computing capacity: not restricted is stronger than restricted
- more money means stronger
- more time means stronger

Defines partial order of attacker (models).
Security in computer networks

**confidentiality**
- message content is confidential
- place • sender / recipient anonymous

**integrity**
- detect forgery
- recipient can prove transmission
- sender can prove transmission
- ensure payment for service

**availability**
- enable communication

end-to-end encryption
mechanisms to protect traffic data

authentication system(s)
sign messages
receipt
during service by digital payment systems

diverse networks;
fair sharing of resources
Multilateral security

- Each party has its particular **protection goals**.

- Each party can **formulate** its protection goals.

- Security conflicts are recognized and compromises **negotiated**.

- Each party can **enforce** its protection goals within the agreed compromise.

**Security with minimal assumptions about others**
Multilateral security (2nd version)

• Each party has its particular goals.

• Each party can formulate its protection goals.

• Security conflicts are recognized and compromises negotiated.

• Each party can enforce its protection goals within the agreed compromise.

Security with minimal assumptions about others
Multilateral security (3rd version)

- Each party has its particular goals.
- Each party can formulate its protection goals.
- Security conflicts are recognized and compromises negotiated.
- Each party can enforce its protection goals within the agreed compromise. As far as limitations of this cannot be avoided, they equally apply to all parties.

Security with minimal assumptions about others
Physical security assumptions

Each technical security measure needs a physical “anchoring” in a part of the system which the attacker has neither read access nor modifying access to.

Range from “computer centre X” to “smart card Y”

What can be expected at best?

Availability of a locally concentrated part of the system cannot be provided against realistic attackers

→ physically distributed system

… hope the attacker cannot be at many places at the same time.

Distribution makes confidentiality and integrity more difficult. But physical measures concerning confidentiality and integrity are more efficient: Protection against all realistic attackers seems feasible. If so, physical distribution is quite ok.
Tamper-resistant casings

Interference: detect judge

Attack: delay delete data (etc.)

Possibility: several layers, shielding
Shell-shaped arrangement of the five basic functions

delay (e.g. hard material),
detect (e.g. sensors for vibration or pressure)

- shield,
- judge

- delete
Tamper-resistant casings

Interference: detect
  judge

Attack: delay
delete data (etc.)

Possibility: several layers, shielding

Problem: validation ... credibility

Negative example: smart cards
  • no detection (battery missing etc.)
  • shielding difficult (card is thin and flexible)
  • no deletion of data intended, even when power supplied
Correspondence between organizational and IT structures
Admission control: communicate with authorized partners only

Access control: subject can only exercise operations on objects if authorized.
Identification of human beings by IT-systems

- What one is:
  - hand geometry
  - finger print
  - picture
  - hand-written signature
  - retina-pattern
  - voice
  - typing characteristics

- has:
  - paper document
  - metal key
  - magnetic-strip card
  - smart card (chip card)
  - calculator

- knows:
  - password, passphrase
  - answers to questions
  - calculation results for numbers

eID-card
New German eID Card

PIN protects access to chip
Identification of IT-systems by human beings

What it *is*
- casing
- seal, hologram
- pollution

What it *knows*
- password
- answers to questions
- calculation results for numbers

Where it *stands*
Identification of IT-systems by IT-systems

What it *knows*

- password
- answers to questions
- calculation results for numbers
- *cryptography*

Wiring *from where*
Password based authentication

- Simple approach

<table>
<thead>
<tr>
<th>Login</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>bone</td>
</tr>
</tbody>
</table>

User → (dog, bone) → Server

- Grant access
- Deny access
Password based authentication

- Simple approach – security problems

Attacker might get access!
Password based authentication

- Enhanced approach using one way (hash) functions
One-way functions – cryptographic hash functions

• One-way function $f$:
  – calculating $f(x)=y$ is easy
  – calculating $f^{-1}(y)=x$ is hard
    • computation / storage
  – open question: Do one-way functions exist?

• Cryptographic hash function $h$
  – might have different properties depending on the use case
  – collision resistance:
    • it is hard to find $x$, $y$ with $h(y)=h(x)$ and $y\neq x$
    • note: $h$ is usually not collision free, because $|h(x)| \ll |x|$
  – preimage resistance / one-way function / secrecy
    • given $h(x)$ it is hard to find $x$
  – second-preimage resistance / weak collision resistance / binding
    • given $x$, $h(x)$ it is hard to find $y$ with $h(y)=h(x)$ and $y\neq x$
  – Note:
    • $h$ is not necessarily a “random extractor”
    • only one of “secrecy” and “binding” can be information theoretic secure
Examples for cryptographic hash functions

- **MD5**
  - Message-Digest Algorithm
  - developed by Ronald Rivest (April 1992)
  - produces 128 bit hash values
  - can process arbitrary long inputs
  - *today MD5 is broken!*

- **SHA-1**
  - Secure Hash Standard
  - published 1993 as FIPS PUB 180 by US NIST
  - produces 160 bit hash values
  - *today SHA-1 is insecure!*

- **SHA-2**
  - set of hash functions, with hash values of 224, 256, 384, 512 bit
  - published 2001 as FIPS PUB 180-2 by NIST
  - **SHA-2 hash functions are believed to be secure**

- **SHA-3**
  - will be the result of the NIST Cryptographic Hash Algorithm Competition started November 2007
  - 3 selection rounds, 5 finalists
  - result expected late 2012
MD5 Hash in the Wild

- United States Cyber Command (USCYBERCOM)
  - mission statement: "USCYBERCOM plans, coordinates, integrates, synchronizes and conducts activities to: direct the operations and defense of specified Department of Defense information networks and; prepare to, and when directed, conduct full spectrum military cyberspace operations in order to enable actions in all domains, ensure US/Allied freedom of action in cyberspace and deny the same to our adversaries."
mission statement: “USCYBERCOM plans, coordinates, integrates, synchronizes and conducts activities to: direct the operations and defense of specified Department of Defense information networks and; prepare to, and when directed, conduct full spectrum military cyberspace operations in order to enable actions in all domains, ensure US/Allied freedom of action in cyberspace and deny the same to our adversaries.”
mission statement:= “USCYBERCOM plans, coordinates, integrates, synchronizes and conducts activities to: direct the operations and defense of specified Department of Defense information networks and; prepare to, and when directed, conduct full spectrum military cyberspace operations in order to enable actions in all domains, ensure US/Allied freedom of action in cyberspace and deny the same to our adversaries.”

MD5(mission statement)= 9ec4c12949a4f31474f299058ce2b22a

(Remember: MD5 is broken → find other interesting mission statements…)}
Password based authentication

- Enhanced approach using one way (hash) functions

![Diagram showing password authentication process.](image_url)
Password based authentication

- Enhanced approach using one way (hash) functions

User → Server

Login Password

(dog,bone)

Server calculates $h$(bone)

Login | Password
--- | ---
... | ...
dog | $h$(bone)
... | ...

Slightly reduced risk, if attacker gets access.

yes → grant access

no → deny access
Remaining problems of password based authentication

based one way functions

• Brute Force attack
  – function $h()$ is public
  – value of $h(x)$ is known to the attacker
  ➔ try all possible values for $x$

Considerations:
  • usually $>> 1$ Mio. $h(x)/s$ on ordinary hardware
  • assumption: password uses only small letters
  • password length = 8

\[
\text{time needed: } \frac{26^8}{1\,000\,000 \cdot 60 \cdot 60} \approx 58h
\]

• first countermeasures:
  – limit false attempts

• first password rules:
  – use a large alphabet (small and capitalised letters, numbers, specials)
  – use a long password
Remaining problems of password based authentication based one way functions

• first password rules:
  – use a large alphabet
    • (small, capitalised letters, numbers, specials)
    • time needed: \( \frac{(26 + 26 + 10 + 30)^8}{1,000,000 \cdot 60 \cdot 60 \cdot 24 \cdot 365.25} \approx 162a \)
  – use a long password

• remaining possible attacks:
  – increase in computation power
    • distributed approach
    • GPU
    • Moore’s law
  – pre-computation:
    • attacker creates lockup table
    • search time (example above):
      \[ \text{ld}((26 + 26 + 10 + 30)^8) < 53 \text{ comparisons} \]
Remaining problems of password based authentication

based one way functions

• remaining possible attack:
  – pre-computation

• countermeasure:
  – salt!
  – \( h(x) \rightarrow h(\text{salt},x) \)
  – salt:
    • long (e.g. 128 bit) random value
    • some part is unique for the system (i.e. 104 bit)
    • some part is randomly chosen by the system for each entry in the password table (i.e. 24 bit)
      – NOT stored at the system
    • verification: iterate over all possible salt values

\[ \Rightarrow \] pre-computation has to be done \textit{for each possible salt}
Remaining problems of password based authentication based one way functions

- remaining possible attack:
  - dictionary attack
  - problem: people do not chose passwords randomly
    - often names, words or predictable numbers are used
    - attacker uses dictionaries for brute force attack
  - prominent program: John the Ripper
    - supports dictionary attacks and password patterns

- possible solutions:
  - enforce password rules
    - consider usability
  - pre-check passwords (e.g. using John)
  - train people to “generate” good passwords
    - Example: sentence \(\rightarrow\) password
    - “This is the password I use for Google mail” \(\rightarrow\) “Titplu4Gm”
City Whistler

• … a new Web 2.0 service
• … for people which like city journeys
• … find cool cities and places like shops, restaurants, hotels etc.
• … information from globe-trotters for globe-trotters
• … they can share their knowledge after secure login
• So that’s wrong?

→ It collects (username, password) and tries to login into other popular services like Gmail, Twitter, eBay, Amazon etc.

password rule: never “reuse” passwords!
Password based authentication

- Simple approach – **security problems**

```
<table>
<thead>
<tr>
<th>Login</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>dog</td>
<td>$h(salt, bone)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

Attacker might get access!
Password based authentication

• security problems

• possible solution:
  – encrypt communication

• remaining problems:
  – not always possible
  – replay attack
Password based authentication

- security problem – replay attack

![Diagram showing password-based authentication with a security problem due to replay attack](image-url)
Password based authentication

- **security problem** – replay attack
- **possible solution**: challenge-response protocol

- tries to ensure *freshness*

- **remaining problems:**
  - Man-in-the-middle attacks
  - parallel protocol runs
Password based authentication

- **security problem** – MITM / parallel protocol runs
Password based authentication

- **security problem** – MITM / parallel protocol runs
- **possible solutions:**
  - disallow parallel login protocol runs for the same user
  - make protocol runs distinguishable
Password based authentication

• possible solution: distinguishable protocol runs

1.1 login request (dog, $r_{dog}$)

2.1 login request (dog, $r_{cat}$)

1.2 random challenge c

2.2 random challenge $c'$

1.2' random challenge $c'$

1.3 response $f(c', secret, r_{dog})$

2.3 response $f(c', secret, r_{dog})$

• will not accept the same $r$ multiple times
Password based authentication

• **security problem** – MITM / parallel protocol runs

• **possible solutions:**
  – disallow parallel login protocol runs for the same user
  – make protocol runs distinguishable

• **remaining security problems:**
  – …

(OK I will stop here – if you are interested in many more problems / solutions I recommend: Colin Boyd, Anish Mathuria: “Protocols for Authentication and Key Establishment”, Springer, 2003.)
Password based authentication

- (non protocol related) security problems:
  - phising, i.e. faked UI for entering secret information
  - today: mostly Internet based attacks
  - but: local attacks possible as well
    - faked login / lock screen
    - solution: “trusted path” / Secure Attention Key

3.2.2.1.1 The TCB [Trusted Computing Base] shall support a trusted communication path between itself and user for initial login and authentication. Communications via this path shall be initiated exclusively by a user. [Department of Defense: “Trusted Computer System Evaluation Criteria”, CSC-STD-001-83, 15. August 1983 – called “Orange Book”]

- well known implementations:
  - Windows: Ctrl+Alt+Del
  - Linux: Ctrl+Alt+Pause
    - could be freely chosen in principle
One time password

• One Time Password
  – only used to authenticate a single transaction

• Advantage
  – abuse of OTP becomes harder for the attacker

• Implementations
  – list of OTPs
    • known from online banking: TAN, iTAN
  – on the fly generated and transmitted over a second channel
    • mTAN
  – time-synchronized (hardware) tokens:
    • token knows a secret s
    • OTP= \( f(s, \text{time}) \)
  – hash chain based
One time password

- OTP Implementations
  - hash chain based
    - Leslie Lamport: “Password Authentication with Insecure Communication”
    - users generates hash chain:
      - $h^n(\ldots h^3(h^2(h^1(\text{password}))))$
    - users sends $h^n()$ as his “password” during register procedure
    - next login user sends $h^{n-1}()$
    - server verifies: $h(h^{n-1}()) = h^n()$
    - server now stores: $h^{n-1}()$
Biometrics for Authentication

- *Physiological* or *behavioural* characteristics (of a human being) are measured and compared with reference values to
  - **verify**, that a given subject is the one it claimed to be
    - claimed “identity” is known to the system by other means
  - **identify**, a subject within a given set of (known) subjects
    - “identity” should be derived from biometrics
    - usually more difficult compared to verification
Biometrics: Physiological / Behavioural Characteristics

- Iris / Retina
- Fingerprint
- DNA
- Hand geometry
- (3D) Face geometry
- Thermography: facial thermograms
- Handwriting: appearance, dynamics of writing
- Voice spectrogram
- Key strokes: dynamics of writing (speed, pressure etc.)
Biometric characteristics: Requirements

- universal: everyone has it
- unique
- stable over time
- measurable
- acceptable
- analysable
- resistant against cloning / faking
Biometrics: Pros and Cons

• Pros:
  – Cannot be divulged or lost/forgotten
  – can be utilised “on the fly”
  – Hard to copy

• Cons:
  – Cannot be renewed
  – Person related data requires special protection (privacy)
  – Invasion (of privacy)
  – Error rate
Biometrics: Pros and Cons

• Pros:
  – Cannot be divulged or lost/forgotten
    • but could be stolen
Demolition Man (1993): Simon Phoenix (Wesley Snipes) escaping from the jail...
Biometrics: Pros and Cons

• Pros:
  – Cannot be divulged or lost/forgotten
    • but could be stolen:
    • could become „unusable“ due to
      – ageing
      – incidents
      – disease
  – can be utilised “on the fly”
    • privacy problems (unnoticeable measurement of Biometrics)
  – Hard to copy
    • depends on the Biometric system used
    • many systems are easy to cheat
Demonstration of Fingerprint Cloning by CCC
Biometrics: Pros and Cons

• Pros:
  – Cannot be divulged or lost/forgotten
    • but could be stolen:
      – http://news.bbc.co.uk/2/hi/asia-pacific/4396831.stm
    • could become „unusable“ due to
      – ageing
      – incidents
      – disease
  – can be utilised “on the fly”
    • privacy problems (unnoticeable measurement of Biometrics)
  – Hard to copy
    • depends on the Biometric system used
    • many systems are easy to cheat
    • ftp://ftp.ccc.de/pub/documentation/Fingerabdruck_Hack/fingerabdruck.mpg
    • cloning of e.g. fingerprints might be in the interest of law enforcement
      – access to biometrically secured devices
Biometric Systems: Types of Failures

- **False Accept Rate (FAR) / False Match Rate (FMR):**
  - Security problem!
- **False Reject Rate (FRR) / False nonmatch Rate (FNR):**
  - Usability / acceptance problem
- **Receiver Operating Characteristic (ROC):**
  - curve of FAR against FRR
- **Equal Error Rate (EER):**
  - rate for FAR=FRR
  - can be seen from the ROC curve
ROC Curve and Security Problems of Biometrics

Low FMR causes high FNR and vice versa!

Figure taken from: Anil Jain, Lin Hong, Sharath Pankanti: Biometric Identification; Communications of the ACM 43/2 (2000) 91-98
Biometric Systems: Types of Failures

- **False Accept Rate (FAR):**
  - **Security problem!**

- **False Reject Rate (FRR):**
  - Usability / acceptance problem

- **Receiver Operating Characteristic (ROC):**
  - curve of FAR against FRR

- **Equal Error Rate (EER):**
  - error rate for FAR=FRR
  - can be seen from the ROC curve

- **Failure To Enroll Rate (FTE):**
  - Usability / acceptance problem

- **Failure To Capture Rate (FTC):**
  - Usability / acceptance problem
Enhanced Security: Multi-biometric Systems

- Multi-sample
- Multi-sensor
- Multi-modal
- Hybrid
- Multi-algorithm
- Multi-instance

- Capacitive
- Optical

- Minutiae
- Structure
- Right eye
- Left eye
Admission control: communicate with authorized partners only

Access control: subject can only exercise operations on objects if authorized.
Computer virus vs. transitive Trojan horse

Access control
Limit spread of attack by as little privileges as possible:
Don‘t grant unnecessary access rights!

No computer viruses, only transitive Trojan horses!
Basic facts about Computer viruses and Trojan horses

Other measures fail:

1. Undecidable if program is a computer virus
   proof (indirect) assumption: decide (•)
   
   \[
   \begin{align*}
   \text{program counter_example} \\
   \text{if decide (counter_example) then no_virus_functionality} \\
   \text{else virus_functionality}
   \end{align*}
   \]

2. Undecidable if program is Trojan horse

Better be too careful!

3. Even known computer viruses are not efficiently identifiable
   self-modification virus scanner

4. Same for: Trojan horses

5. Damage concerning data is not ascertainable afterwards
   function inflicting damage could modify itself
Further problems

1. Specify exactly what IT system should do and what it *must not* do.

2. Prove *total correctness* of implementation.

3. Are all *covert channels* identified?
Design and realize IT system as *distributed* system, such that a limited number of attacking computers cannot inflict significant damage.
Aspects of distribution

physical distribution
distributed control and implementation structure

distributed system:

no entity has a global view on the system
Security in distributed systems

Trustworthy terminals

Trustworthy only to user to others as well

Ability to communicate

Availability by redundancy and diversity

Cryptography

Confidentiality by encryption
Integrity by message authentication codes (MACs) or digital signatures
Availability

Infrastructure with the least possible complexity of design

Connection to completely diverse networks
  • different frequency bands in radio networks
  • redundant wiring and diverse routing in fixed networks

Avoid bottlenecks of diversity
  • e.g. radio network needs same local exchange as fixed network,
  • for all subscriber links, there is only one transmission point to the long distance network
Achievable protection goals:

- **confidentiality**, called **concealment**
- **integrity** (= no *undetected* unauthorized modification of information), called **authentication**

Unachievable by cryptography:

- **availability** – at least not against strong attackers
Symmetric encryption system

*Key generation*

- Random number \( r \)
- Secret key \( k := \text{gen}(r) \)

*Encryption*

- Encryption function \( k(x) \)
- Ciphertext \( S := \text{enc}(k, x) \)

*Decryption*

- Decryption function \( k^{-1}(k(x)) \)
- Plaintext \( x := \text{dec}(k, S) = \text{dec}(k, \text{enc}(k, x)) \)

Domain of trust

- NS\(\text{A}: \text{Bad Aibling} \)
- Law enforcement: wiretapping interface

Secret area

- Local computer
  - Hardware (HW): no side-channels
  - Operating system: Windows 95/98/ME/CE/XP Home, MacOS 9.x: all programs
- Area of attack
  - NSA: Bad Aibling
  - Law enforcement: wiretapping interface

Opaque box with lock; 2 identical keys
Example: Vernam cipher (=one-time pad)

- **Example:** Vernam cipher (=one-time pad)
- **Schlüsselgenerierung**
- **Verschlüsselung**
- **Entschlüsselung**
- **ciphertext**
- **plaintext**
- **random number**
- **secret key**
- **opaque box with lock; 2 identical keys**
Key exchange using symmetric encryption systems

key exchange centers

X

Y

Z

NSA:
Key Escrow
Key Recovery

$k_{AX}(k_1)$ $k_{AY}(k_2)$ $k_{AZ}(k_3)$  $k_{BX}(k_1)$ $k_{BY}(k_2)$ $k_{BZ}(k_3)$

key $k = k_1 + k_2 + k_3$

$k$(messages)

participant A

participant B
Sym. encryption system: Domain of trust key generation

Symmetric encryption system:

- **Encryption:** $x \rightarrow k(x)$
- **Decryption:** $k^{-1}(k(x)) \rightarrow x$

**Domain of trust:**
- Key generation
- Encrypter, decrypter, or key exchange center

**Area of attack:** Secret area
Asymmetric encryption system

more detailed notation

Domain of trust

plaintext $x$ -> encryption $c(x)$ -> ciphertext $S$

Area of attack

$c$ (encryption key, publicly known)

d (decryption key, kept secret)

$S := \text{enc}(c, x, r')$

$Opaque box with spring lock; 1 key$
Key distribution using asymmetric encryption systems

1. A registers his public encryption key $c_A$ (possibly anonymously).

2. B asks the key register $R$ for the public encryption key of $A$.

3. B gets the public encryption key $c_A$ of $A$ from $R$, certified by $R$'s signature.

$A$ registers his public encryption key $c_A$ (possibly anonymously).

$B$ asks the key register $R$ for the public encryption key of $A$.

$B$ gets the public encryption key $c_A$ of $A$ from $R$, certified by $R$'s signature.
Symmetric authentication system

**Show-case with lock; 2 identical keys**

**Domain of trust**
- Random number $r$
- $k := \text{gen}(r)$

**Domain of trust**
- Secret key $k$
- Plaintext $x$
- $\text{MAC} := \text{code}(k, x)$
- Test: $\text{MAC} = \text{code}(k, x)$
- “Pass” or “Fail”

**Area of attack**
- Plaintext and test result $x$,
- Secret area

**Encode**
- Plaintext $x$
- $\text{code}$(plaintext)
- $\text{MAC} := \text{code}(k, x)$
-

**Key generation**
- Random number $r$
- $k := \text{gen}(r)$

**More detailed notation**
- $x$
- $k(x)$
- $\text{MAC} = \text{code}(k, x)$
Digital signature system

more detailed notation

Domain of trust (no confidentiality needed)
plaintext with signature and test result

show-case with lock; 1 key
Key distribution using digital signature systems

1. A registers $t_A$ the key for testing his signature (possibly anonymously).

2. $B$ requests the key for testing the signature of $A$ from key register $R$.

3. $B$ receives key $t_A$ for testing the signature of $A$ from $R$, certified by the signature of $R$.

message from $A$, $s_A$(message from $A$)
Key generation

generation of a random number \( r \) for the key generation:

XOR of

\[ r_1, \text{ created in device,} \]
\[ r_2, \text{ delivered by producer,} \]
\[ r_3, \text{ delivered by user,} \]
\[ r_n, \text{ calculated from keystroke intervals.} \]
Needham-Schroeder-Protocol using Symmetric encryption

- from 1978

- goals:
  - key freshness:
    - key is „fresh“, i.e. a newly generated one
  - key authentication:
    - key is only known to Alice and Bob (and maybe some trusted third party)

- preconditions:
  - a trusted third party $T$
  - shared term secret keys between Alice (resp. Bob) and the trusted third party:
    - $k_{AT}$, $k_{BT}$
Needham-Schroeder-Protocol using Symmetric encryption

key exchange center

1. A, B, N_A

2. k_A(T)(N_A, B, k_{AB}, k_{BT}(k_{AB}, A))

3. k_{BT}(k_{AB}, A)

4. k_{AB}(N_B)

5. k_{AB}(N_B^{-1})

• Problem?
Needham-Schroeder-Protocol using Asymmetric encryption

- from 1978

- goals:
  - key freshness:
    - key is “fresh“, i.e. a newly generated one
  - key authentication:
    - key is only known to Alice and Bob

- preconditions:
  - public encryption keys of Alice $c_A$ and Bob $c_B$ known to each other
Needham-Schroeder-Protocol using Asymmetric encryption

1. $c_B(N_A, A)$
2. $c_A(N_A, N_B)$
3. $c_B(N_B)$

$k_{AB} = \text{KDF}(N_A, N_B)$

$k_{AB}$ (messages)

participant A

participant B

- Problem?
Comments on key exchange

Whom are keys assigned to?

1. individual participants     asymmetric systems
2. pair relations               symmetric systems
3. groups                       –

How many keys have to be exchanged?

\[ n \text{ participants} \]

\[ \text{asymmetric systems: } n \text{ per system} \]
\[ \text{symmetric systems: } n \cdot (n-1) \]

When are keys generated and exchanged?

Security of key exchange limits security available by cryptography:

execute several initial key exchanges
Goal/success of attack

a) key (total break)

b) procedure equivalent to key (universal break)

c) individual messages,
   e.g. especially for authentication systems
      c1) one selected message (selective break)
      c2) any message (existential break)
Types of attack

- a) passive
  - a1) ciphertext-only attack
  - a2) known-plaintext attack

- b) active
  - (according to encryption system; asym.: either b1 or b2; sym.: b1 or b2)
    - b1) signature system: plaintext → ciphertext (signature)
       (chosen-plaintext attack)
    - b2) encryption system: ciphertext → plaintext
       (chosen-ciphertext attack)

Adaptivity
- not adaptive
- adaptive

Severity

Criterion: action
- passive attacker ≠ observing attacker
- active attacker ≠ modifying attacker
Basic facts about “cryptographically strong” (1)

If no security against computationally unrestricted attacker:

1) using of keys of constant length $\ell$.
   - attacker algorithm can always try out all $2^\ell$ keys
     (breaks asym. encryption systems and sym. systems in known-plaintext attack).
   - requires an exponential number of operations
     (too much effort for $\ell > 100$).

→ the best that the designer of encryption systems can hope for.

2) complexity theory:
   - mainly delivers asymptotic results
   - mainly deals with “worst-case”-complexity

→ useless for security; same for “average-case”-complexity.

goal: problem is supposed to be difficult almost everywhere, i.e. except for an infinitesimal fraction of cases.
   - security parameter $\ell$ (more general than key length; practically useful)
   - if $\ell \to \infty$, then probability of breaking $\to 0$.
   - hope: slow, fast

If no security against computationally unrestricted attacker:
Basic facts about “cryptographically strong” (2)

3) 2 classes of complexity:
   - En-/decryption: easy = polynomial in $\mathcal{L}$
   - Breaking: hard = not polynomial in $\mathcal{L}$ $\approx$ exponential in $\mathcal{L}$

   Why?
   a) harder than exponential is impossible, see 1).
   b) self-contained: substituting polynomials in polynomials gives polynomials.
   c) reasonable models of calculation (Turing-, RAM-machine) are polynomially equivalent.

   For practice polynomial of high degree would suffice for runtime of attacker algorithm on RAM-machine.

4) Why assumptions on computational restrictions, e.g., factoring is difficult?
   - Complexity theory cannot prove any useful lower limits so far.
   - Compact, long studied assumptions!

5) What if assumption turns out to be wrong?
   a) Make other assumptions.
   b) More precise analysis, e.g., fix model of calculation exactly and then examine if polynomial is of high enough degree.

6) Goal of proof: If attacker algorithm can break encryption system, then it can also solve the problem which was assumed to be difficult.
Security classes of cryptographic systems

1. attacker assumed to be computationally unrestricted
2. cryptographically strong
3. well analyzed
4. somewhat analyzed
5. kept secret
<table>
<thead>
<tr>
<th>Security</th>
<th>System Type</th>
<th>Concealment</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Theoretic</td>
<td>Vernam cipher (one-time pad)</td>
<td>1</td>
<td>authentication codes</td>
</tr>
<tr>
<td>Cryptographically Strong</td>
<td>pseudo one-time pad with $s^2 \mod n$ generator</td>
<td>3</td>
<td>CS</td>
</tr>
<tr>
<td>Well Analyzed</td>
<td>mathematics</td>
<td>5</td>
<td>system with $s^2 \mod n$ generator</td>
</tr>
<tr>
<td></td>
<td>chaos</td>
<td>8</td>
<td>RSA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>DES</td>
</tr>
</tbody>
</table>
Hybrid cryptosystems (1)

Combine:
- from asymmetric systems: easy key distribution
- from symmetric systems: efficiency (factor 100 ... 10000, SW and HW)

How?
use asymmetric system to distribute key for symmetric system

Encryption:

\[ A \xrightarrow{M} B \]

get \( c_B \)
choose \( k \)
\( c_B(k), k(M) \)
decrypt \( k \) with \( d_B \)
decrypt \( M \) with \( k \)
Hybrid cryptosystems (2)

Even more efficient: part of $M$ in first block

If $B$ is supposed also to use $k$: append $s_A(B,k)$

Authentication: $k$ authorized and kept secret
### Information-theoretically secure encryption (1)

“Any ciphertext $S$ may equally well be any plaintext $x$”

<table>
<thead>
<tr>
<th>ciphertext</th>
<th>key $k$</th>
<th>plaintext $x$</th>
<th>ciphertext</th>
<th>key $k$</th>
<th>plaintext $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>$k$</td>
<td>$x$</td>
<td>$S$</td>
<td>$k$</td>
<td>$x$</td>
</tr>
<tr>
<td>00</td>
<td></td>
<td>00</td>
<td>00</td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td>01</td>
<td>01</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>10</td>
<td>10</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>11</td>
<td>11</td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>

**secure cipher**

**insecure cipher**
Information-theoretically secure encryption (2)

“Any ciphertext S may equally well be any plaintext x”

<table>
<thead>
<tr>
<th>ciphertext</th>
<th>key</th>
<th>plaintext</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>k</td>
<td>x</td>
</tr>
<tr>
<td>00</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>

secure cipher

```
example : Vernam cipher mod 2
x = 00 01 00 10
⊕ k = 10 11 01 00
S = 10 10 01 10
```

insecure cipher

subtraction of one key bit mod 4 from 2 plaintext bits
Different probability **distributions** – how do they fit?

<table>
<thead>
<tr>
<th>ciphertext</th>
<th>key</th>
<th>plaintext</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>k</td>
<td>x</td>
</tr>
<tr>
<td>00</td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>

Unevenly distributed plaintexts enciphered with equally distributed keys yield **equally distributed ciphertexts**.

**equally distributed** **equally distributed** **unevenly distributed**
Different probability **distributions** – how do they fit?

<table>
<thead>
<tr>
<th>ciphertext</th>
<th>key</th>
<th>plaintext</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>$k$</td>
<td>$x$</td>
</tr>
</tbody>
</table>

Equally distributed ciphertexts deciphered with **equally distributed** keys can yield **unevenly distributed** plaintexts, iff ciphertexts and keys are **not** independently distributed, i.e., the ciphertexts have been calculated using the plaintext and the key.
Vernam cipher (one-time pad)

All characters are elements of a group $G$.

Plaintext, key and ciphertext are character strings.

For the encryption of a character string $x$ of length $n$, a randomly generated and secretly exchanged key $k = (k_1, ..., k_n)$ is used.

The $i^{th}$ plaintext character $x_i$ is encrypted as

$$S_i := x_i + k_i$$

It can be decrypted with

$$x_i := S_i - k_i.$$  

Evaluation:  
1. secure against adaptive attacks  
2. easy to calculate  
3. but key is very long
Keys have to be very long for information-theoretical security

\( K \) is the set of keys, 
\( X \) is the set of plaintexts, and 
\( S \) is the set of ciphertexts, which appear at least once.

\[ |S| \geq |X| \] otherwise it can’t be decrypted (fixed \( k \))

\[ |K| \geq |S| \] so that any ciphertext might as well be any plaintext (fixed \( x \))

therefore \[ |K| \geq |X| \].

If plaintext cleverly coded, it follows that:

The length of the key must be at least the length of the plaintext.
1. Definition for information-theoretical security
(all keys are chosen with the same probability)

\[
\forall S \in S \exists \text{ const } \in \mathbb{N} \ \forall x \in X: |\{k \in K| k(x) = S\}| = \text{ const}.
\]  
(1)

The a-posteriori probability of the plaintext \(x\) is \(P(x|S)\), after the attacker got to know the ciphertext \(S\).

2. Definition

\[
\forall S \in S \ \forall x \in X: P(x|S) = P(x).
\]  
(2)

Both definitions are equivalent (if \(P(x) > 0\)):

According to Bayes:

\[
P(x|S) = \frac{P(x) \cdot P(S|x)}{P(S)}
\]

Therefore, (2) is equivalent to

\[
\forall S \in S \ \forall x \in X: P(S|x) = P(S).
\]  
(3)

We show that this is equivalent to

\[
\forall S \in S \ \exists \text{ const}' \in \mathbb{R} \ \forall x \in X: P(S|x) = \text{ const}'.
\]  
(4)
Proof

(3)⇒(4) is clear with \( \text{const}' := P(S) \).

(4)⇒(3): Conversely, we show \( \text{const}' = P(S) \):

\[
P(S) = \sum_x P(x) \cdot \text{const}'
\]

\[
\forall S \in S \exists \text{const}' \in \mathbb{R} \quad \forall x \in X: P(S|x) = \text{const}' \quad (4)
\]

\[
\text{const}' \cdot \sum_x P(x)
\]

\[
\forall S \in S \exists \text{const} \in \mathbb{N} \quad \forall x \in X: \left| \{ k \in \mathcal{K} \mid k(x) = S \} \right| = \text{const}. \quad (1)
\]

(4) is already quite the same as (1): In general holds

\[
P(S|x) = P(\{k \mid k(x) = S\}),
\]

and if all keys have the same probability,

\[
P(S|x) = \left| \{ k \mid k(x) = S \} \right| / |\mathcal{K}|.
\]

Then (4) is equivalent (1) with

\[
\text{const} = \text{const}' \cdot |\mathcal{K}|.
\]
Symmetric authentication systems (1)

Key distribution:
like for symmetric encryption systems

Simple example (view of attacker)

The outcome of tossing a coin (Head (H) or Tail (T)) shall be sent in an authenticated fashion:

<table>
<thead>
<tr>
<th>$k$</th>
<th>$x, MAC$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>H - T -</td>
</tr>
<tr>
<td>01</td>
<td>H - - T</td>
</tr>
<tr>
<td>10</td>
<td>- H T -</td>
</tr>
<tr>
<td>11</td>
<td>- H - T</td>
</tr>
</tbody>
</table>

Security: e.g. attacker wants to send $T$.

a) blind: get caught with a probability of 0.5

b) seeing: e.g. attacker gets $H,0$  $\Rightarrow$  $k \in \{00, 01\}$

still both, $T,0$ and $T,1$, have a probability of 0.5
Symmetric authentication systems (2)

Definition “Information-theoretical security”
with error probability $\varepsilon$:

$\forall x, \text{MAC}$ (that attacker can see)
$\forall y \neq x$ (that attacker sends instead of $x$)
$\forall \text{MAC}'$ (where attacker chooses the one with the highest probability fitting $y$)

$W(k(y) = \text{MAC}' | k(x) = \text{MAC}) \leq \varepsilon$

(probability that MAC' is correct if one only takes the keys $k$ which are still possible under the constraint of $(x,\text{MAC})$ being correct.)

Improvement of the example:

a) $2\sigma$ key bits instead of 2: $k = k_1 k_1^* ... k_{\sigma} k_{\sigma}^*$
   $\text{MAC} = \text{MAC}_1, ... , \text{MAC}_{\sigma}$; $\text{MAC}_i$ calculated using $k_i k_i^*$
   $\Rightarrow$ error probability $2^{-\sigma}$

b) $l$ message bits: $x^{(1)}, \text{MAC}^{(1)} = \text{MAC}_{1}^{(1)}, ... , \text{MAC}_{\sigma}^{(1)}$
   $\vdots$
   $x^{(l)}, \text{MAC}^{(l)} = \text{MAC}_{1}^{(l)}, ... , \text{MAC}_{\sigma}^{(l)}$
Limits:

\[ \sigma \text{-bit-MAC} \Rightarrow \text{error probability} \geq 2^{-\sigma} \]
(guess MAC)

\[ \sigma \text{-bit-key} \Rightarrow \text{error probability} \geq 2^{-\sigma} \]
(guess key, calculate MAC)

still clear: for an error probability of \(2^{-\sigma}\), a \(\sigma\)-bit-key is too short, because \(k(x) = \text{MAC}\) eliminates many values of \(k\).

Theorem: you need \(2\sigma\)-bit-key
(for succeeding messages \(\sigma\) bits suffice, if recipient adequately responds on authentication “errors”)

Possible at present: \(\approx 4\sigma \cdot \log_2(\text{length}(x))\)
(Wegman, Carter)

much shorter as one-time pad
About cryptographically strong systems (1)

Mathematical secrets:

(to decrypt, to sign ...)

\( p, q, \) prime numbers

Public part of key-pair:

(to encrypt, to test ...)

\( n = p \cdot q \)

\( p, q \) big, at present \( \approx \mathcal{L} = 500 \) up to 2000 bit

(theory: \( \mathcal{L} \rightarrow \infty \))

Often: special property

\( p \equiv q \equiv 3 \mod 4 \)

(the semantics of “\( \equiv \ldots \mod \)” is:
\( a \equiv b \mod c \) iff \( c \) divides \( a-b \),
putting it another way: dividing \( a \) and \( b \)
by \( c \) leaves the same remainder)
application: $s^2$-mod-$n$-generator, 
GMR and many others, 
e.g., only well analyzed systems like RSA

(significant alternative: only “discrete logarithm”,
based on number theory, too, similarly well analyzed)

necessary: 1. factoring is difficult
2. to generate $p,q$ is easy
3. operations on the message with $n$ alone, you can only invert using $p$, $q$
Factoring

clear: in NP ⇒ but difficulty cannot be proved yet

complexity at present

\[ L(n) = e^{c \cdot 3 \sqrt{\ln(n) \cdot (\ln \ln(n))^2}} \]

\[ \approx e^{3 \sqrt{l}} \]

practically up to 155 decimal digits in the year 1999
174 decimal digits in the year 2003
200 decimal digits in the year 2005
232 decimal digits in the year 2010
240 decimal digits in the year 2019 (www.crypto-world.com/FactorRecords.html)

(notice:
∃ faster algorithms, e.g., for 2^r ± 1, but this doesn’t matter)

assumption: factoring is hard

(notice: If an attacker could factor, e.g., every 1000^{th} n, this would be unacceptable.)
Factoring assumption

\( \forall \text{ PPA } \mathcal{F} \) (probabilistic polynomial algorithm, which tries to factor)

\( \forall \) polynomials \( Q \)

\( \exists \ L \ \forall \ \ell \geq L : \) (asymptotically holds:)

If \( p, q \) are random prime numbers of length \( \ell \) and \( n = p \cdot q \):

\[
W(\mathcal{F}(n) = (p, q)) \leq \frac{1}{Q(\ell)}
\]

(probability that \( \mathcal{F} \) truly factors decreases faster as \( \frac{1}{\text{any polynomial}} \)).

trustworthy ??

the best analyzed assumption of all available
1. Are there enough prime numbers? (important also for factoring assumption)

\[
\frac{\pi(x)}{x} \approx \frac{1}{\ln(x)}
\]

\(\pi(x)\) number of the prime numbers \(\leq x\)

“prime number theorem”

\[\Rightarrow \text{up to length } \ell \text{ more than every } \ell^{th}.\]

And \(\approx\) every \(2^{nd}\) \(\equiv 3 \mod 4\) “Dirichlet’s prime number theorem”

2. Principle of search:

repeat

choose random number \(p \equiv 3 \mod 4\)

test whether \(p\) is prime

until \(p\) prime
3. Primality tests:

(notice: trying to factor is much too slow)

probabilistic; “Rabin-Miller”

special case $p \equiv 3 \mod 4$ :

$$p \text{ prime} \quad \Rightarrow \quad \forall \ a \neq 0 \mod p : \quad a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p}$$

$$p \text{ not prime} \quad \Rightarrow \quad \text{for at most } \frac{1}{4} \text{ of } a \text{'s} : \quad a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p}$$

$\Rightarrow$ test this for $m$ different, independently chosen values of $a$,

error probability $\leq \frac{1}{4^m}$

(doesn’t matter in general)
$\mathbb{Z}_n$: ring of residue classes mod $n \doteq \{0, \ldots, n-1\}$

- $+, -, \cdot$ fast

- Exponentiation “fast” (square & multiply)

Example: $7^{26} = 7^{(11010)_2}$; from left

- $\gcd$ (greatest common divisor) fast in $\mathbb{Z}$ (Euclidean Algorithm)
Calculating with and without $p,q$ (2)

$Z^*_n$: multiplicative group

$a \in Z^*_n \iff \gcd (a,n) = 1$

- Inverting is fast (extended Euclidean Algorithm)

Determine to $a,n$ the values $u,v$ with

$$a \cdot u + n \cdot v = 1$$

Then:

$$u \equiv a^{-1} \mod n$$

Example: $3^{-1} \mod 11$?

$$11 = 3 \cdot 3 + 2$$

$$3 = 1 \cdot 2 + 1$$

$$= -11 + 4 \cdot 3$$

$$1 = 1 \cdot 3 - 1 \cdot (11 - 3 \cdot 3)$$

$$\Rightarrow 3^{-1} \equiv 4 \mod 11$$
Calculating with and without $p, q$ (3)

Number of elements of $\mathbb{Z}_n^*$

The Euler $\Phi$-Function is defined as

$$\Phi(n) := \left| \{a \in \{0, \ldots, n-1\} \mid \gcd(a,n)=1\} \right|,$$

whereby for any integer $n \neq 0$ holds: $\gcd(0,n)=\left| n \right|$.

It immediately follows from both definitions, that

$$\left| \mathbb{Z}_n^* \right| = \Phi(n).$$

For $n = p \cdot q$, $p, q$ prime and $p \neq q$ we can easily calculate $\Phi(n)$:

$$\Phi(n) = (p-1) \cdot (q-1)$$

gcd $\neq 1$ have the numbers 0, then $p, 2p, \ldots, (q-1)p$ and $q, 2q, \ldots, (p-1)q$, and these $1+(q-1)+(p-1) = p+q-1$ numbers are for $p \neq q$ all different.
Calculating with and without $p, q$ (4)

Relation between $\mathbb{Z}_n \leftrightarrow \mathbb{Z}_p, \mathbb{Z}_q$:

**Chinese Remainder Theorem (CRA)**

\[
x \equiv y \mod n \iff x \equiv y \mod p \land x \equiv y \mod q
\]

since

\[
n | (x-y) \iff p | (x-y) \land q | (x-y)
\]

\[n = p \cdot q, \ p, q \text{ prime, } p \neq q\]

$\Rightarrow$ To calculate $f(x) \mod n$, at first you have to calculate $\mod p$, $q$ separately.

\[
y_p := f(x) \mod p
\]

\[
y_q := f(x) \mod q
\]
Calculating with and without $p, q$ (5)

Compose?

extended Euclidean: $u \cdot p + v \cdot q = 1$

$y := (u \cdot p) \cdot y_q + (v \cdot q) \cdot y_p$

\[
\begin{array}{c|c|c}
 & \text{mod } p & \text{mod } q \\
\hline
u \cdot p & 0 & 1 \\
\hline
v \cdot q & 1 & 0 \\
\hline
y & 0 \cdot y_q + 1 \cdot y_p & 1 \cdot y_q + 0 \cdot y_p \\
\hline
\equiv y_p & \equiv y_q
\end{array}
\]

CRA
RSA - asymmetric cryptosystem


Key generation

1) Choose two prime numbers \( p \) and \( q \) at random as well as stochastically independent, with \( |p| \approx |q| = \ell, \ p \neq q \)

2) Calculate \( n := p \cdot q \)

3) Choose \( c \) with \( 3 \leq c < (p-1)(q-1) \)

4) Calculate \( d \) using \( p, q, c \) as multiplicative inverse of \( c \mod \Phi(n) \)
   \[
   c \cdot d \equiv 1 \pmod{\Phi(n)}
   \]

5) Publish \( c \) and \( n \).

En- / decryption

Exponentiation with \( c \) respectively \( d \) in \( \mathbb{Z}_n \)

Proposition: \( \forall m \in \mathbb{Z}_n \) holds: \( (m^c)^d \equiv m^c \cdot d \equiv (m^d)^c \equiv m \pmod{n} \)
Proof (1)

\[ c \cdot d \equiv 1 \pmod{\Phi(n)} \iff \exists k \in \mathbb{Z}: c \cdot d - 1 = k \cdot \Phi(n) \iff \exists k \in \mathbb{Z}: c \cdot d = k \cdot \Phi(n) + 1 \]

Therefore \[ m^c \cdot d \equiv m^k \cdot \Phi(n) + 1 \pmod{n} \]

Using the Theorem of Fermat \[ \forall m \in \mathbb{Z}_n^*: m^{\Phi(n)} \equiv 1 \pmod{n} \]

it follows for all \( m \) coprime to \( p \) \[ m^{p-1} \equiv 1 \pmod{p} \]

Because \( p-1 \) is a factor of \( \Phi(n) \), it holds \[ m^{k \cdot \Phi(n) + 1} \equiv_p m^{k \cdot (p-1)(q-1) + 1} \equiv_p m \cdot (m^{p-1})^k \cdot (q-1) \equiv_p m \]
Proof (2)

Holds, of course, for \( m \equiv_p 0 \). So we have it for all \( m \in \mathbb{Z}_p \).

Same argumentation for \( q \) gives

\[
m^k \cdot \phi(n) + 1 \equiv_q m
\]

Because congruence holds relating to \( p \) as well as \( q \), according to the CRA, it holds relating to \( p \cdot q = n \).

Therefore, for all \( m \in \mathbb{Z}_n \)

\[
m^c \cdot d \equiv m^k \cdot \phi(n) + 1 \equiv m \quad \text{(mod } n)\]

**Attention:**
There is (until now ?) **no** proof
RSA is easy to break \( \Rightarrow \) to factor is easy
Semantic Security

Let’s play a game:
A challenger flips a coin, and the adversary guesses the outcome
For \( b=0,1 \) define experiments EXP(0) and EXP(1) as:

- \( W_0 := \) [event that EXP(0) = 1] (wrong guess)
- \( W_1 := \) [event that EXP(1) = 1] (correct guess)

\[
\Pr[W_0] - \Pr[W_1] \in [0,1]
\]

- \( E \) is called **semantically secure** if for all efficient algorithms \( A \),
  \( \text{Advantage}_{SS}[A,E] := |\Pr[W_0] - \Pr[W_1]| \in [0,1] \)
- \( E \) is called **semantically secure** if for all efficient algorithms \( A \),
  \( \text{Advantage}_{SS}[A,E] \) is negligible (\( \sim 0 \))
Semantic Security of plain RSA?

Let’s play a game:
A challenger flips a coin, and the adversary guesses the outcome
For \( b=0,1 \) define experiments \( \text{EXP}(0) \) and \( \text{EXP}(1) \) as:

- No: Attacker can always encrypt and compare!
  \( \Rightarrow \) indeterministic encryption necessary!
  add random number…

(Based on slide from Prof. Thorsten Strufe)
Calculating with and without $p,q$ (6)

squares and roots

$$\text{QR}_n := \{ x \in Z_n^* \mid \exists y \in Z_n^* : y^2 \equiv x \mod n \}$$

$x$ : “quadratic residue”

$y$ : “root of $x$”

$-y$ is also a root

but attention: e.g. mod 8

$$1^2 \equiv 1 \quad 3^2 \equiv 1 \quad 4 \quad 7^2 \equiv 1 \quad 5^2 \equiv 1$$

$\text{QR}_n$ multiplicative group:

$$x_1, x_2 \in \text{QR}_n \Rightarrow x_1 \cdot x_2 \in \text{QR}_n : (y_1 y_2)^2 = y_1^2 y_2^2 = x_1 x_2$$

$$x_1^{-1} \in \text{QR}_n : (y_1^{-1})^2 = (y_1^2)^{-1} = x_1^{-1}$$
Calculating with and without $p, q$ (7)

**squares and roots mod $p$, prime:**

$\mathbb{Z}_p$ field

$\Rightarrow$ as usual $\leq 2$ roots

$x \neq 0, p \neq 2$ : 0 or 2 roots

$\Rightarrow |\text{QR}_p| = \frac{p - 1}{2}$

(square function is $2 \rightarrow 1$)

<table>
<thead>
<tr>
<th>$x$</th>
<th>$0$</th>
<th>$1$</th>
<th>$2$</th>
<th>$\ldots$</th>
<th>$\frac{p-1}{2}$</th>
<th>$-\frac{p-1}{2}$</th>
<th>$\ldots$</th>
<th>$-2$</th>
<th>$-1$ = $p - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^2$</td>
<td>$0$</td>
<td>$1$</td>
<td>$4$</td>
<td>$\ldots$</td>
<td></td>
<td>$\ldots$</td>
<td></td>
<td>$4$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

**Jacobi symbol**

\[
\left( \frac{x}{p} \right) := \begin{cases} 
1 & \text{if } x \in \text{QR}_p \\
-1 & \text{else}
\end{cases} 
\quad \text{(for } x \in \mathbb{Z}_p^*)
\]
Calculating with and without $p,q$ (8)

Continuation squares and roots mod $p$, prime:

Euler criterion: \[
\left( \frac{x}{p} \right) \equiv x^{\frac{p-1}{2}} \mod p
\]

(i.e. fast algorithm to test whether square)

Proof using little Theorem of Fermat: \( x^{p-1} \equiv 1 \mod p \)

co-domain ok: \( x^2 \in \{\pm 1\} \), because \( \left( x^2 \right)^{\frac{p-1}{2}} \equiv 1 \)

\( x \) square:
\[
\left( \frac{x}{p} \right) = 1 \Rightarrow x^2 \equiv (y^2)^{\frac{p-1}{2}} \equiv y^{p-1} \equiv 1
\]

\( x \) nonsquare: The $\frac{p-1}{2}$ solutions of \( x^2 \equiv 1 \) are the squares. So no nonsquare satisfies the equation.

Therefore: \( x^2 \equiv -1 \).
The $s^2$-mod-$n$-Pseudo-random Bitstream Generator (PBG)

Idea: short initial value (seed) $\rightarrow$ long bit sequence (should be random from a polynomial attacker’s point of view)

Scheme:

- key and initial value
- generation of key and initial value
- PBG
- real random number
- security-parameter

Requirements:

- gen and PBG are efficient
- PBG is deterministic ($\Rightarrow$ sequence reproducible)
- secure: no probabilistic polynomial test can distinguish PBG-streams from real random streams

secret area

length poly($\ell$)

long bitstream $b_0 \ b_1 \ b_2 \ ...$
**s^2-mod-n-generator**

**Method**

- **key value:** \( p,q \) prime, big, \( \equiv 3 \mod 4 \)
  \( n = p \cdot q \)
- **initial value (seed):** \( s \in \mathbb{Z}_n^* \)
- **PBG:**
  \[
  s_0 := s^2 \mod n \\
  s_{i+1} := s_i^2 \mod n \\
  b_i := s_i \mod 2
  \]

... (last bit)

**Example:** \( n = 3 \cdot 11 = 33 \), \( s = 2 \)

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_i : )</td>
<td>4</td>
<td>16</td>
<td>25</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>( b_i : )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

16^2 \mod 33 = 8 \cdot 32 = 8 \cdot (-1) = 25

25^2 = (-8)^2 \equiv 64 \equiv 31

31^2 = (-2)^2 = 4

Note: length of period no problem with big numbers
(Blum / Blum / Shub 1983 / 86)
Purpose: application as symmetric encryption system: “Pseudo one-time pad”

Compare: one-time pad: add long real random bit stream with plaintext
Pseudo one-time pad: add long pseudo-random stream with plaintext

Scheme:

key generation = generation of key and initial value

security-parameter

real random number

secret key = key and initial value

encryption:
create \( b_0, b_1, b_2, \ldots \), add

plaintext ciphertext plaintext

= \( x_0x_1x_2 \ldots \)

decryption:
create \( b_0, b_1, b_2, \ldots \), add

plaintext

= \( x_0 \oplus b_0, x_1 \oplus b_1, \ldots \)
Idea:
If no probabilistic polynomial test can distinguish pseudo-random streams from real random streams, then the pseudo one-time pad is as good as the one-time pad against polynomial attacker.

(Else the attack is a test !)

Construction works with any good PBG
Proof of security of pseudo one-time pad: another approach

- Prerequisite:
  - Unpredictable PRNG, which cannot be distinguished from real randomness
- We known:
  - One-time pad (XOR with truly random bit string) is secure

Proof intuition:

1. $m_0, m_1 \xleftarrow{} K$
2. $c \xleftarrow{} m_0 \oplus \text{PRNG}(k)$
3. $b' \approx 1$
4. $m_0, m_1 \xleftarrow{} \{0,1\}^n$
5. $c \xleftarrow{} m_0 \oplus r$
6. $b' \approx 1$
7. $r \xleftarrow{} \{0,1\}^n$
8. $c \xleftarrow{} m_1 \oplus r$
9. $b' \approx 1$

(Based on slide from Prof. Thorsten Strufe)
Calculating with and without $p, q$ (9)

squares and roots mod $p \equiv 3 \mod 4$

- extracting roots is easy: given $x \in \text{QR}_p$

\[ w := x^{\frac{p+1}{4}} \mod p \text{ is root} \]

proof:

1. $p \equiv 3 \mod 4 \Rightarrow \frac{p+1}{4} \in \mathbb{N}$

2. $w^2 = x^{\frac{p+1}{2}} = x^{\frac{p-1}{2}+1} = x^{\frac{p-1}{2}} \cdot x = 1 \cdot x
\]

Euler, $x \in \text{QR}_p$

In addition: $w \in \text{QR}_p$ (power of $x \in \text{QR}_p$) → extracting roots iteratively is possible

\[ \left( \frac{-1}{p} \right) \equiv (-1)^{\frac{p-1}{2}} \equiv (-1)^{\frac{4r+2}{2}+2r+1} = (-1)^{2r+1} = -1 \]

$p = 4r+3$

$\Rightarrow -1 \not\in \text{QR}_p$

$\Rightarrow$ of the roots $\pm w$: $-w \not\in \text{QR}_p$ (otherwise $-1 = (-w) \cdot w^{-1} \in \text{QR}_p$)
Calculating with and without $p,q$ (10)

squares and roots mod $n$ using $p,q$
(usable as secret operations)

• testing whether square is simple \( (n = p \cdot q, \ p,q \text{ prime, } p\neq q) \)

\[
x \in \text{QR}_n \iff x \in \text{QR}_p \land x \in \text{QR}_q
\]

Chinese Remainder Theorem

proof: \( \Rightarrow \) \[
x \equiv w^2 \mod n \ \Rightarrow \ x \equiv w^2 \mod p \land x \equiv w^2 \mod q
\]

\( \Leftarrow \) \[
x \equiv w_p^2 \mod p \land x \equiv w_q^2 \mod q
\]

\[
w := \text{CRA}(w_p, w_q)
\]
then \[
w \equiv w_p \mod p \land w \equiv w_q \mod q
\]
using the Chinese Remainder Theorem for
\[
w^2 \equiv w_p^2 \equiv x \mod p \land w^2 \equiv w_q^2 \equiv x \mod q
\]
we have
\[
w^2 \equiv x \mod n
\]
Continuation squares und roots mod $n$ using $p,q$

$x \in \mathbb{QR}_n \Rightarrow x$ has exactly 4 roots

(mod $p$ and mod $q$: $\pm w_p, \pm w_q$.
therefore the 4 combinations according to the Chinese Remainder Theorem)

• extracting a root is easy ($p, q \equiv 3 \mod 4$)
determine roots $w_p, w_q$ mod $p, q$

\[
\begin{align*}
  w_p &:= x^{\frac{p+1}{4}} \\
  w_q &:= x^{\frac{q+1}{4}}
\end{align*}
\]

combine using CRA
s^2-mod-n-generator as asymmetric encryption system

chosen ciphertext-plaintext attack

plaintext $x = x_0 x_1 x_2 ...$

$S$ random initial value

encryption: create
$s_0 s_1 s_2 ...$
$b_0 b_1 b_2 ...$
add

$c(x) = x_0 \oplus b_0,$
$x_1 \oplus b_1, ...$
$x_k \oplus b_k, 1, (s_{k+1})^2$

ciphertext

secret area

security-parameter

public key = modulus

real random number

private key = factors

$n$

key generation

plaintext $c(x)$

decryption: create
$s_{k+1} s_k s_{k-1} ... s_1 s_0$
$b_0 b_1 b_2 ...$
add

plaintext $x = x_0, x_1, x_2 ..., 1 0$
Calculating with and without \( p,q \) (12)

Continuation squares und roots mod \( n \) using \( p,q \)

Jacobi symbol

\[
\begin{pmatrix} \frac{x}{n} \end{pmatrix} := \begin{pmatrix} \frac{x}{p} \end{pmatrix} \cdot \begin{pmatrix} \frac{x}{q} \end{pmatrix}
\]

So:
\[
\begin{pmatrix} \frac{x}{n} \end{pmatrix} = \begin{cases} +1 & \text{if } x \in \text{QR}_p \land x \in \text{QR}_q \lor x \notin \text{QR}_p \land x \notin \text{QR}_q \\ -1 & \text{if } \text{“cross-over”} \end{cases}
\]

So : \( x \in \text{QR}_n \) \( \Rightarrow \begin{pmatrix} \frac{x}{n} \end{pmatrix} = 1 \)

\( \Leftrightarrow \) does not hold
Calculating with and without $p, q$ (13)

continuation squares und roots mod $n$ using $p, q$

to determine the Jacobi symbol is easy

e.g. $p \equiv q \equiv 3 \mod 4$

\[
\left( \frac{-1}{n} \right) = \left( \frac{-1}{p} \right) \cdot \left( \frac{-1}{q} \right) = (-1) \cdot (-1) = 1
\]

but $-1 \notin \text{QR}_n$, because $\notin \text{QR}_{p,q}$
squares and roots mod $n$ without $p,q$

- extracting roots is difficult: provably so difficult as to factor
  a) If someone knows 2 significantly different roots of an $x \mod n$, then he can definitely factor $n$.
    (i.e. $w_1^2 \equiv w_2^2 \equiv x$, but $w_1 \not\equiv \pm w_2 \Rightarrow n \not\mid (w_1 \pm w_2)$)

  proof: $n \mid w_1^2 - w_2^2 \Rightarrow n \mid (w_1 + w_2)(w_1 - w_2)$

  $p$ in one factor, $q$ in the other

  $\Rightarrow \gcd(w_1 + w_2, n)$ is $p$ or $q$
Calculating with and without $p,q$ (15)

Continuation squares and roots mod $n$ without $p,q$

b) Sketch of “factoring is difficult $\Rightarrow$ extracting a root is difficult”
proof of “factoring is easy $\iff$ extracting a root is easy”

So assumption: $\exists \mathcal{W} \in \text{PPA}:$ algorithm extracting a root

to show: $\exists \mathcal{F} \in \text{PPA}:$ factoring algorithm

structure

program $\mathcal{F}$

subprogram $\mathcal{W}$

[black box]

begin

... call $\mathcal{W}$

... polynomially often

... call $\mathcal{W}$

... end.
Calculating with and without \( p, q \) (16)

to b)

\[ \mathcal{F} : \quad \text{input } n \]

repeat forever

choose \( w \in \mathbb{Z}_n^* \) at random, set \( x := w^2 \)

\( w' := \mathcal{W}(n, x) \)

test whether \( w' \equiv \pm w \), if so factor according to a) break

- to determine the Jacobi symbol is easy

(if \( p \) and \( q \) unknown: use quadratic law of reciprocity)

but note: If \( \left( \frac{x}{n} \right) = 1 \), determine whether \( x \in \text{QR}_n \) is difficult

(i.e. it does not work essentially better than to guess)

\[ \textbf{QRA} = \text{quadratic residuosity assumption} \]
Security of the $s^2$-mod-$n$-generator (1)

unpredictability to the left will do

$\forall P \in \text{PPA} \quad \{\text{predictor for } b_0\}$

$\forall \text{ constants } \delta, 0 < \delta < 1 \quad \{\text{frequency of the “bad” } n \}$

$\forall t \in \mathbb{N} : \quad \{\text{degree of the polynomial}\}$

if $\mathcal{L} (= |n|)$ sufficiently big it holds: for all keys $n$ except of at most a $\delta$-fraction

$W(b_0=\mathcal{P}(n,b_1b_2...b_k) | s \in \mathbb{Z}_n^* \text{ random}) < \frac{1}{2} + \frac{1}{\mathcal{L}^t}$
Security of the $s^2$-mod-$n$-generator (2)

Proof: Contradiction to QRA in 2 steps

Assumption: $s^2$-mod-$n$-generator is weak, i.e. there is a predictor $P$, which guesses $b_0$ with $\varepsilon$-advantage given $b_1 \ b_2 \ b_3 \ ...$

Step 1: Transform $P$ in $P^*$, which to a given $s_1$ of $\text{QR}_n$ guesses the last bit of $s_0$ with $\varepsilon$-advantage.

Given $s_1$.
Generate $b_1 \ b_2 \ b_3 \ ...$ with $s^2$-mod-$n$-generator, apply $P$ to that stream. $P$ guesses $b_0$ with $\varepsilon$-advantage. That is exactly the result of $P^*$.

Step 2: Construct using $P^*$ a method $R$, that guesses with $\varepsilon$-advantage, whether a given $s^*$ with Jacobi symbol $+1$ is a square.

Given $s^*$.
Set $s_1 := (s^*)^2$.
Apply $P^*$ to $s_1$. $P^*$ guesses the last bit of $s_0$ with $\varepsilon$-advantage, where $s^*$ and $s_0$ are roots of $s_1$; $s_0 \in \text{QR}_n$.
Therefore $s^* \in \text{QR}_n \iff s^* = s_0$
The last bit $b^*$ of $s^*$ and the guessed $b_0$ of $s_0$ suffice to guess correctly, because

1) if $s^* = s_0$, then $b^* = b_0$

2) to show: if $s^* \neq s_0$, then $b^* \neq b_0$

if $s^* \neq s_0$ because of the same Jacobi symbols, it holds

$$s^* \equiv -s_0 \mod n$$

therefore $s^* = n - s_0$ in $\mathbb{Z}$

$n$ is odd, therefore $s^*$ and $s_0$ have different last bits

The constructed $\mathcal{R}$ is in contradiction to QRA.

Notes:
1) You can take $O(\log(\mathcal{L}))$ random bits in place of (last) 1 bit per squaring.
2) There is a more difficult proof that $s^2$-mod-$n$-generator is secure under the factoring assumption.
Requirements for a PBG:

“strongest” requirement: PBG passes each probabilistic Test $T$ with polynomial running time.

pass = streams of the PBG cannot be distinguished from real random bit stream with significant probability by any probabilistic test with polynomial running time.

probabilistic test with polynomial running time = probabilistic polynomial-time restricted algorithm that assigns to each input of $\{0,1\}^*$ a real number of the interval $[0,1]$. (value depends in general on the sequence of the random decisions.)

Let $\alpha_m$ be the average (with respect to an even distribution) value, that $T$ assigns to a random $m$-bit-string.
PBG passes $\mathcal{T}$ iff

For all $t > 0$, for sufficiently big $\mathcal{L}$ the average (over all initial values of length $\mathcal{L}$), that $\mathcal{T}$ assigns to the poly($\mathcal{L}$)-bit-stream generated by the PBG, is in $\alpha_{\text{poly}(\mathcal{L})} \pm 1/\mathcal{L}^t$

To this “strongest” requirement, the following 3 are equivalent (but easier to prove):

For each generated finite initial bit string, of which any (the rightmost, leftmost) bit is missing, each polynomial-time algorithm $\mathcal{P}$ (predictor) can “only guess” the missing bit.

Idea of proof: From each of these 3 requirements follows the “strongest”

- easy: construct test from predictor
- hard: construct predictor from test
Security of PBGs more precisely (3)

Proof (indirect): Construct predictor $\mathcal{P}$ from the test $\mathcal{T}$.

For a $t>0$ and infinitely many $\ell$ the average (over all initial values of length $\ell$), that $\mathcal{T}$ assigns to the generated $\text{poly}(\ell)$-bit-string of the PBG is (e.g. above) $\alpha_{\text{poly}(\ell)} \pm 1/\ell^t$. Input to $\mathcal{T}$ a bit string of 2 parts: $j+k=\text{poly}(\ell)$

**real random**

$A=\{r_1 ... r_j \, r_{j+1} \, b_1 ... b_k\}$ are assigned a value closer to $\alpha_{\text{poly}(\ell)}$

$B=\{r_1 ... r_j \, b_0 \, b_1 ... b_k\}$ are assigned a value more distant to $\alpha_{\text{poly}(\ell)}$, generated by PBG e.g. higher

Predictor for bit string $b_1 ... b_k$ constructed as follows:

$\mathcal{T}$ on input $\{r_1 ... r_j \, 0 \, b_1 ... b_k\}$ estimate $\alpha^0$

$\mathcal{T}$ on input $\{r_1 ... r_j \, 1 \, b_1 ... b_k\}$ estimate $\alpha^1$

Guess $b_0 = 0$ with probability of $1/2 + 1/2 (\alpha^0 - \alpha^1)$

(more precisely: L. Blum, M. Blum, M. Shub: A simple unpredictable Pseudo-Random Number Generator; SIAM J. Comput. 15/2 (May 1986) page 375f)
Naive insecure use of RSA

RSA as asymmetric encryption system

Code message (if necessary in several pieces) as number $m < n$

Encryption of $m$: $m^c \mod n$

Decryption of $m^c$: $(m^c)^d \mod n = m$

RSA as digital signature system

Renaming: $c \rightarrow t, d \rightarrow s$

Signing of $m$: $m^s \mod n$

Testing of $m, m^s$: $(m^s)^t \mod n = m \ ?$
RSA as asymmetric encryption system: naive

Key generation:
- \( p, q \) prime numbers
- \( n := p \cdot q \)
- \( c \) with \( \gcd(c, (p-1)(q-1)) = 1 \)
- \( d \equiv c^{-1} \mod (p-1)(q-1) \)

Encryption:
- \( x \) → \( c(x) \equiv x^c \mod n \)

Decryption:
- \( (c(x))^d \equiv (x^c)^d \equiv x \mod n \)

Secret area:
- Random number
- Private key, kept secret
- \( d, n \)

Publicly known:
- \( c, n \)
- Encryption key

Random number:
- Random number

plaintext → ciphertext → plaintext
RSA as asymmetric encryption system: example

key generation:
- \( p, q \) = 3, 11
- \( n \) = 33
- \( c \) = 13 with \( \gcd(13, 20) = 1 \)
- \( d \) = 17

encryption
- \((-2)^{13} \equiv (-2)^5 \cdot (-2)^5 \cdot (-2)^3 \equiv 1 \cdot 1 \cdot (-8) \equiv 25\)

decryption
- \(25^{17} \equiv (-8)^{17} \equiv 64^8 \cdot (-8) \equiv (-2)^8 \cdot (-8) \equiv (-2)^5 \cdot (-2)^5 \cdot (-2) \equiv 1 \cdot 1 \cdot (-2) \equiv 31\)

random number
secret area
encryption key, publicly known
decryption key, kept secret
plaintext
ciphertext
random number

random number
RSA as digital signature system: naive

key generation:
- $p, q$ prime numbers
- $n := p \cdot q$
- $t$ with $\gcd(t, (p-1)(q-1)) = 1$
- $s \equiv t^{-1} \mod (p-1)(q-1)$

"decryption" $(s(x))^t = (x^s)^t \equiv x \mod n$

"encryption" $x^s \mod n$

secret area

random number

$x, s(x)$

$x, s(x)$

$x, s(x), t(x, s(x))$

text with signature and test result

random number

random number

text with signature and test result

key to test the signature, publicly known

key to sign, kept secret

text
Attack on digital signature with RSA naive

\[(x^s)^t\]  
\[\equiv x\quad \text{message wanted}\]

\[\left((x^s \cdot y\right)^t\]  
\[\equiv x \cdot y^t\quad \text{chosen message } y\]

\[\left((x^s \cdot y\right)^t\]  
\[\equiv x^s \cdot y\quad \text{divide by } y, \text{ get } x^s\]
Attack on encryption with RSA naive

\[(x^c)^d \equiv x\]

(ciphertext intercepted)

\[((x \cdot y)^c)^d = x^c \cdot y^c\]

(calculated from y by the attacker)

\[\equiv x \cdot y\]

(divide by y, get x)
Transition to Davida’s attacks

simple version of Davida’s attack: (against RSA as signature system)

1. Given
   
   \[
   \text{Sig}_1 = m_1^s \\
   \text{Sig}_2 = m_2^s \\
   \Rightarrow \quad \text{Sig} := \text{Sig}_1 \cdot \text{Sig}_2 = (m_1 \cdot m_2)^s 
   \]
   New signature generated!
   (Passive attack, \( m \) not selectable.)

2. Active, desired \( \text{Sig} = m^s \)
   
   Choose any \( m_1; \quad m_2 := m \cdot m_1^{-1} \)
   
   Let \( m_1, m_2 \) be signed.
   
   Further as mentioned above.

3. Active, more skillful (Moore)
   
   “Blinding”: choose any \( r \)
   
   \[
   m_2 := m \cdot r^t \\
   m_2^s = m^s \cdot r^t \cdot s = m^s \cdot r \\
   \approx m^s = \text{Sig}
   \]
Defense against Davida’s attacks using a collision-resistant hash function

**h()**: collision-resistant hash function

1.) asymmetric encryption system

plaintext messages have to fulfill redundancy predicate

\[ m, \text{redundancy} \Rightarrow \text{test if } h(m) = \text{redundancy} \]

2.) digital signature system

Before signing, \( h \) is applied to the message

signature of \( m \) = \( (h(m))^s \mod n \)

\[ \text{test if } h(m) = ((h(m))^s)^t \mod n \]

**Attention**: There is no proof of security (so far?)
Faster calculation of the secret operation

mod \( p, q \) separately:
\[
y^d \equiv w
\]

once and for all:
\[
d_p := c^{-1} \mod p-1 \Rightarrow (y^{d_p})^c \equiv y \mod p
\]
\[
d_q := c^{-1} \mod q-1 \Rightarrow (y^{d_q})^c \equiv y \mod q
\]

every time:
\[
\text{set } w := \text{CRA} \left( y^{d_p}, y^{d_q} \right)
\]

proof:
\[
\Rightarrow w^c = \begin{cases} 
(y^{d_p})^c \equiv y \mod p \\
(y^{d_q})^c \equiv y \mod q
\end{cases}
\]
\[
\Rightarrow w^c \equiv y \mod n
\]

How much faster?

complexity exponentiation: \( \approx \ell^3 \)

complexity 2 exponentiations of half the length: \( \approx 2 \cdot \left( \frac{\ell}{2} \right)^3 = \frac{\ell^3}{4} \)

complexity CRA: 2 multiplications \( \approx 2 \cdot \ell \)

1 addition \( \approx \ell \)

So: \( \approx \text{Factor 4} \)
Summary of PBG and motivation of GMR

Reminder:

$s^2$-mod-$n$-generator is secure against passive attackers for arbitrary distributions of messages

- reason for arrow: random number’ in picture asymmetric encryption systems
- memorize term: probabilistic encryption

Terms:

- one-way function
- one-way permutation
  - one-way = nearly nowhere practically invertible
  - variant: invertible with secret (trap door)

Motivation:

active attack on $s^2$-mod-$n$-generator as asymmetric encryption system
Scheme of security proofs (1)

Passive attacker

Alg.1: get to know something about the plaintext (or provide signature, respectively)

Call \rightarrow Result

Alg.2: solve the number theoretic problem

Result

Alg.3: get secret key

Ciphertext

Attacked person

- Choose random number
- Generate key
- Publish a part of the key, if appropriate

Constructive proof often
Seemingly, there are no provably secure cryptosystems against adaptive active attacks. A constructive security proof seems to be a game with fire.
**GMR – signature system**

Shafi Goldwasser, Silvio Micali, Ronald Rivest:
A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks;

**Main ideas**

1) Map a randomly chosen reference $\mathcal{R}$, which is only used once.
2) Out of a set of collision-resistant permutations (which are invertible using a secret) assign to any message $m$ one permutation.

$$
\mathcal{R} \xleftrightarrow{\mathcal{F}_{n,m}^{-1}} (\mathcal{R}) \xleftrightarrow{\mathcal{F}_{n,m}} (\text{Sig}_m^{\mathcal{R}})
$$
Consequence

“variation of $m$” (active attack) now means also a
“variation of $R$” – a randomly chosen reference, that is unknown to the attacker when he chooses $m$.

Problems

1) securing the originality of the randomly chosen reference
2) construction of the collision-resistant permutations (which are invertible only using the secret) which depend on the messages

Solution of problem 2

Idea Choose 2 collision-resistant permutations $f_0, f_1$ (which are invertible only using the secret) and compose $F_{n,m}$ by $f_0, f_1$.

{for simplicity, we will write $f_0$ instead of $f_{n,0}$ and $f_1$ instead of $f_{n,1}$}

Def. Two permutations $f_0, f_1$ are called collision-resistant iff it is difficult to find any $x, y, z$ with $f_0(x) = f_1(y) = z$

Note Proposition: collision-resistant $\Rightarrow$ one-way

Proof (indir.): If $f_i$ isn’t one-way: 1) choose $x$; 2) $f_{1-i}(x) = z$; 3) $f_{i-1}(z) = y$
GMR – signature system (2)

Construction:
For $m = b_0 b_1 ... b_k$ ($b_0, ..., b_k \in \{0,1\}$) let

$$F_{n,m} := f_{b_0} \circ f_{b_1} \circ ... \circ f_{b_k}$$

$$F_{n,m}^{-1} := f_{b_k}^{-1} \circ ... \circ f_{b_1}^{-1} \circ f_{b_0}^{-1}$$

Signing: $R \xrightarrow{f_{b_0}^{-1}} f_{b_0}^{-1}(R) \xrightarrow{f_{b_1}^{-1}} ... \xrightarrow{f_{b_k}^{-1}} f_{b_k}^{-1}(...(f_{b_0}^{-1}(R))\ldots) =: \text{Sig}_m R$

Testing: $\text{Sig}_m \xrightarrow{f_{b_k}} f_{b_k}(\text{Sig}_m) \xrightarrow{f_{b_{k-1}}} ... \xrightarrow{f_{b_0}} f_{b_0}(...(f_{b_k}(\text{Sig}_m))\ldots) = R \ ?$

Example:

$\text{Sig}_m R \xrightarrow{f_0} \xrightarrow{f_1} \xrightarrow{f_1} \xrightarrow{f_1} \xrightarrow{f_1} R$
Problem: intermediate results of the tests are valid signatures for the start section of the message \( m \).

Idea: coding the message prefix free.

Def. A mapping \(<\cdot\>: \text{M} \rightarrow \text{M}\) is called prefix free iff \( \forall m_1, m_2 \in \text{M}: \forall b \in \{0,1\}^+: \langle m_1 \rangle b \neq \langle m_2 \rangle \) \(<\cdot\)> injective.

Example for a prefix free mapping:
\[ 0 \rightarrow 00 ; \ 1 \rightarrow 11 ; \ \text{end identifier} \ 10 \]

Prefix-free encoding should be efficient to calculate both ways.
To factor is difficult (1)

**Theorem:** If factoring is difficult, then collision-resistant permutation pairs exist.

**Proof:**
secret: $p \cdot q = n$; $p \equiv_8 3$ und $q \equiv_8 7$ (Blum numbers)

it holds:
\[
\left(\frac{-1}{n}\right) = 1 \\
\left(\frac{2}{n}\right) = -1
\]

\[f_0(x) := \begin{cases} 
  x^2 \mod n, & \text{if } x < \frac{n}{2} \\
  -x^2 \mod n, & \text{else}
\end{cases}
\]

\[f_1(x) := \begin{cases} 
  (2x)^2 \mod n, & \text{if } x < \frac{n}{2} \\
  -(2x)^2 \mod n, & \text{else}
\end{cases}
\]

Domain: \( \{ x \in \mathbb{Z}_n^* \mid \left(\frac{x}{n}\right) = 1, \ 0 < x < \frac{n}{2} \} \)
To factor is difficult (2)

to show: 1) Permutation = one-to-one mapping with co-domain = domain

2) To calculate the inverse is easy using $p, q$

3) If there is a fast collision finding algorithm, then there is a fast algorithm to factor.

$x^2 \equiv_n -(2y)^2$ cannot hold, since $(2y)^2 \in \text{QR}_n$. Therefore $x^2 \equiv_n (2y)^2 \Rightarrow (x+2y)(x-2y) \equiv_n 0$.

Because $\left(\frac{x}{n}\right) = 1$ and $\left(\frac{\pm 2y}{n}\right) = -1$ it follows that $x \not\equiv_n \pm 2y$

Therefore $\gcd(x \pm 2y, n)$ provides a non-trivial factor of $n$, i.e. $p$ or $q$. 
Solution of problem 1 (1)

Tree of references

signature system 1

signature system 2

The attacker gets to know $\mathcal{R}_i$ only after choosing $m_i$. 

\begin{align*}
\text{generate (\approx sign)} \\
\text{signature system 1} \\
\text{signature system 2}
\end{align*}

\begin{align*}
\text{test} \\
\mathcal{F}_{n,<r_0r_1>} (\text{Sig } r_j) &= r_j \\
\mathcal{F}_{n,<\mathcal{R}_i>} (\text{Sig } \mathcal{R}_i) &= \mathcal{R}_i \\
\mathcal{F}_{n',<m_i>} (\text{Sig } m_i) &= \mathcal{R}_i
\end{align*}
Proposition If the permutation pairs are collision resistant, then the adaptive active attacker can’t sign any message with GMR.

Proof A forged signature leads either to a collision in the tree of references (contradiction) or to an additional legal signature. So the attacker has inverted the collision-resistant permutation. With this ability he could generate collisions (contradiction).

Example:
In the proof you dispose the “Oracle” (the attacked person) by showing that the attacker can generate „half“ the tree from the bottom or (exclusive) “half” the tree from the top with the same probability distribution as the attacked person.

**Lesson:**
randomly chosen references each used only once (compare one-time-pad) make adaptive active attacks ineffective

→ arrow explained (random number $z'$) in figure signature system
GMR signature system

Key generation:
\[ p, p' \equiv 3 \mod 8 \]
\[ q, q' \equiv 7 \mod 8 \]
\[ r \in \mathbb{Z}_n : n := p \cdot q \]
\[ n' := p' \cdot q' \]

---

Test M-signature
R-signature
and K-signatures generate tree of references once and for all or for each message one “branch”

\[ m, s(m) \]

“pass” or “fail”

plaintext with signature and test result

Key for testing of signature; publicly known

\[ n, n', r \in \mathbb{Z}_n \]

Key for signing; kept secret

\[ p, q, p', q', r \in \mathbb{Z}_n \]

Random number

\[ z \]

---

\[ MSig = F_{präf(m)}^{-1}(R_i) \]
\[ RSig = F_{präf(R_i)}^{-1}(r_i) \]
\[ KSig = F_{präf(r_i \cdot r_1)}^{-1}(r_{i-1}), \ldots \]
\[ F_{präf(r_1 \cdot r_1)}^{-1}(r_1) \]
Symmetric Cryptosystem DES (IBM, 1975)

64-bit block plaintext

initial permutation

round 1

round 2

round 16

64-bit key (only 56 bits in use)

generation of a key for each of the 16 rounds

64-bit-block ciphertext
One round

Feistel ciphers    self-inverse!

\[ L_{i-1} \]

\[ R_{i-1} \]

\[ L_i = R_{i-1} \]

\[ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \]
Why does decryption work?

Encryption round $i$

$L_{i-1} \rightarrow R_{i-1}$

Decryption round $i$

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \rightarrow L_i$

$L_i = R_{i-1}$

Decryption

$\begin{align*}
L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(L_i, K_i) &= L_{i-1} \\
L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(R_{i-1}, K_i) &= L_{i-1}
\end{align*}$

replace $L_i$ by $R_{i-1}$
Encryption function $f$

Expansion

Use key

Mixing

Terms
- Substitution-permutation networks
- Confusion - diffusion
Generation of a key for each of the 16 rounds

64-bit key
(only 56 bits in use)

PC-1

C₀

D₀

LS₁

C₁

D₁

LS₂

C₂

D₂

...

C₁₆

D₁₆

PC-2

K₁

PC-2

K₂

PC-2

K₁₆

choose 48 of the 56 bits for each key of the 16 rounds
The complementation property of DES

\[ \text{DES}(\bar{k}, \bar{x}) = \overline{\text{DES}(k, x)} \]
One round

\[ L_i = R_{i-1} \]
\[ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \]
Encryption function $f$

- $R_{i-1}$
- $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_5 \rightarrow S_6 \rightarrow S_7 \rightarrow S_8$
- $P$
- $f(R_{i-1}, K_i)$

- Original, as $0 \oplus 0 = 1 \oplus 1$ and $1 \oplus 0 = 0 \oplus 1$

- Complement
Generalization of DES

1.) $56 \Rightarrow 16 \cdot 48 = 768$ key bits

2.) variable substitution boxes

3.) variable permutations

4.) variable expansion permutation

5.) variable number of rounds
**Goal:**
Strengthen DES by increasing key length

Let \( E : K \times M \rightarrow M \) be a block cipher (DES)

Define \( 3E : K^3 \times M \rightarrow M \) as

\[
3E((k_1, k_2, k_3), m) = E(k_1, D(k_2, E(k_3, m)))
\]

For 3DES: key-size = 3\( \times 56 = 168 \) bits. 3\( \times \)slower than DES.

Why not \( E(E(E(m))) \)? ... What if: \( k_1 = k_2 = k_3 \)?
Simple attack feasible in time \( \approx 2^{118} \)
Meet-in-the-middle attack (no double DES?)

Define $2E((k_1, k_2), m) = E(k_1, E(k_2, m))$

Idea: test if $E(m) = D(c)$

Step 1: build table of encryptions $E(k,m)$
Step 2: for all $k \in \{0,1\}^{56}$ do:
  - test if $D(k, c)$ is in 2nd column.

![Diagram](image)

29.10.2020
Complexity of Meet-in-the-Middle

Time = $2^{56} \log(2^{56}) + 2^{56} \log(2^{56}) < 2^{63} \ll 2^{112}$, space $\approx 2^{56}$

Same attack on 3DES: Time = $2^{118}$, space $\approx 2^{56}$
Stream cipher
  synchronous
  self synchronizing

Block cipher
  Modes of operation:
    Simplest: ECB (electronic codebook)
      each block separately
    But:     concealment: block patterns identifiable
              authentication: blocks permutable
Main problem of ECB

- ECB
- e.g. 64 bits with DES
- block borders
- plaintext blocks
- ciphertext blocks

same plaintext blocks $\xrightarrow{\text{ECB}}$ same ciphertext blocks

Telefax example ($\rightarrow$ compression is helpful)
Electronic Codebook (ECB)

Encryption

Key

Plaintext block $n$

Ciphertext block $n$

Decryption

Key

Plaintext block $n$

Bit error

$n + 1$ $n$
Cipher Block Chaining (CBC)

All lines transmit as many characters as a block comprises:

- Addition mod appropriately chosen modulus
- Subtraction mod appropriately chosen modulus

If error on the line:
Resynchronization after 2 blocks, but block borders have to be recognizable.

Key

Memory for ciphertext block $n-1$

Encryption

Plaintext block $n$

Bit error

Self synchronizing

Decryption

Ciphertext block $n$

Plaintext block $n$
Cipher Block Chaining (CBC) (2)

All lines transmit as many characters as a block comprises

- Addition mod appropriately chosen modulus
- Subtraction mod appropriately chosen modulus

1 modified plaintext bit ⇒ from there on completely different ciphertext

memory for ciphertext block \( n-1 \)

memory for ciphertext block \( n-1 \)

bit error

\( n+2 \) \( n+1 \) \( n \)

plaintext block \( n \)

key

encryption

\( n+2 \) \( n+1 \) \( n \)

ciphertext block \( n \)

key

decryption

\( n+2 \) \( n+1 \) \( n \)

plaintext block \( n \)

useable for authentication ⇒ use last block as MAC
CBC for authentication

plaintext block $n$ → encryption $+$ key $→$ ciphertext block $n$ $→$ memory for ciphertext block $n-1$ $→$ ciphertext block $n$ $→$ encryption $+$ key $→$ ciphertext block $n$ $→$ memory for ciphertext block $n-1$ $→$ ciphertext block $n$ $→$ last block $→$ comparison $→$ ok ?

plaintext block $n$ $→$ encryption $+$ key $→$ ciphertext block $n$ $→$ memory for ciphertext block $n-1$ $→$ ciphertext block $n$ $→$ encryption $+$ key $→$ ciphertext block $n$ $→$ memory for ciphertext block $n-1$ $→$ ciphertext block $n$ $→$ last block $→$ comparison $→$ ok ?
Pathological Block cipher

plaintext block (length $b$)

<table>
<thead>
<tr>
<th>$x_1$ $x_2$ $x_3$ \ldots $x_{b-1}$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>secure</td>
<td></td>
</tr>
<tr>
<td>$S_1$ $S_2$ $S_3$ \ldots $S_{b-1}$</td>
<td>1</td>
</tr>
</tbody>
</table>

plaintext block (length $b-1$)

<table>
<thead>
<tr>
<th>$x_1$ $x_2$ $x_3$ \ldots $x_{b-1}$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$ $S_2$ $S_3$ \ldots $S_{b-1}$</td>
<td>1</td>
</tr>
</tbody>
</table>

ciphertext block (length $b$)

<table>
<thead>
<tr>
<th>$x_1$ $x_2$ $x_3$ \ldots $x_{b-1}$</th>
<th>1</th>
</tr>
</thead>
</table>

insecure

ciphertext block (length $b-1$)

<table>
<thead>
<tr>
<th>$x_1$ $x_2$ $x_3$ \ldots $x_{b-1}$</th>
<th>0</th>
</tr>
</thead>
</table>
Cipher FeedBack (CFB)

- $b$ Block length
- $a$ Length of the output unit, $a \leq b$
- $r$ Length of the feedback unit, $r \leq b$
- $\oplus$ Addition mod appropriately chosen modulus
- $\ominus$ Subtraction mod appropriately chosen modulus

Block length
- $a$ Length of the output unit, $a \leq b$
- $r$ Length of the feedback unit, $r \leq b$
- $\oplus$ Addition mod appropriately chosen modulus
- $\ominus$ Subtraction mod appropriately chosen modulus

**Symmetric; self synchronizing**

**Diagram:**
- Shift register
- Encryption
- Choose or complete
- Plaintext
- Ciphertext

**Formulas:**
- $a \oplus a \oplus a \oplus a$
- $b \oplus b \oplus b \oplus b$
- $r \oplus r \oplus r \oplus r$

**Key:**
- Input
- Output

**Plaintext:**
- Input
- Ciphertext

**Ciphertext:**
- Output
- Plaintext
Cipher FeedBack (CFB) (2)

- **b** Block length
- **a** Length of the output unit, \( a \leq b \)
- **r** Length of the feedback unit, \( r \leq b \)

- Addition mod appropriately chosen modulus
- Subtraction mod appropriately chosen modulus

**Diagram:**
- **Key** → Encryption → Choose or Complete → Plaintext
- **Ciphertext** → Choose or Complete → Encryption → Key

**Notes:**
- **Shifting Register:**
  - Length: \( n \)
  - Modulus: \( n+1 \) or \( n+2 \)
- **Encryption:**
  - Modulo: \( a \)
  - Self-synchronizing
CFB for authentication

Encryption

Shift register

Key

Choose or complete

Plaintext stream

Comparison

Ok?

Last content of the shift register encrypted

Choose or complete

Choose
Output FeedBack (OFB)

\[ b \] Block length
\[ a \] Length of the output unit, \( a \leq b \)
\[ r \] Length of the feedback unit, \( r \leq b \)
\[ \oplus \] Addition mod appropriately chosen modulus
\[ \ominus \] Subtraction mod appropriately chosen modulus

\[ \text{key} \rightarrow \text{encryption} \]
\[ \text{choose} \]
\[ \text{choose or complete} \]
\[ a \]
\[ \text{plaintext} \rightarrow \text{ciphertext} \]

\[ \text{symmetric; synchronous} \]
\[ \text{Pseudo-one-time-pad} \]
Counter Mode (CTR)

- **Counter (CTR)** encryption
- **Counter Value**
- **Increment Counter**
- **Key**
- **Plaintext**
- **Ciphertext**
- **Symmetric; synchronous**
- **Pseudo-one-time-pad**

Diagram:
- Key → Encryption → Counter Value → Increment Counter → Encryption
- Plaintext → XOR → Ciphertext → XOR → Plaintext
Plain Cipher Block Chaining (PCBC)

All lines transmit as many characters as a block comprises

- Addition mod appropriately chosen modulus, e.g. 2
- Subtraction mod appropriately chosen modulus, e.g. 2
- Any function, e.g. addition mod 2

\( h \) - Block length

memory for plaintext block \( n-1 \)

memory for ciphertext block \( n-1 \)

memory for ciphertext block \( n-1 \)

memory for plaintext block \( n-1 \)

key

key

plaintext ciphertext plaintext

block \( n \)

block \( n \)

block \( n \)

Addition mod appropriately chosen modulus, e.g. 2

Subtraction mod appropriately chosen modulus, e.g. 2

Any function, e.g. addition mod 2

plaintext block \( n \)
Output Cipher FeedBack (OCFB)

- **b** Block length
- **a** Length of the output unit, \( a \leq b \)
- **r** Length of the feedback unit, \( r \leq b \)
- Addition mod appropriately chosen modulus
- Subtraction mod appropriately chosen modulus
- Any function

### Diagram

- **Encryption** block
- **Shift Register**
- **Choose or Complete**
- **Plaintext**
- **Ciphertext**

**Key Points**

- Symmetric; synchronous
- Block length \( b \)
- Output unit length \( a \leq b \)
- Feedback unit length \( r \leq b \)
- Modulus arithmetic
- Any function

**Equations**

\[
\begin{align*}
\text{Key} &
\rightarrow \\
\text{Encryption} &
\rightarrow \\
\text{Shift Register} &
\rightarrow \\
\text{Choose or Complete} &
\rightarrow \\
\text{Plaintext} &
\rightarrow \\
\text{Ciphertext} &
\rightarrow 
\end{align*}
\]
## Properties of the operation modes

<table>
<thead>
<tr>
<th></th>
<th>ECB</th>
<th>CBC</th>
<th>PCBC</th>
<th>CFB</th>
<th>OFB</th>
<th>OCFB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Utilization of indeterministic block cipher</strong></td>
<td>+ possible</td>
<td></td>
<td></td>
<td></td>
<td>- impossible</td>
<td></td>
</tr>
<tr>
<td><strong>Use of an asymmetric block cipher results in</strong></td>
<td></td>
<td>+ asymmetric stream cipher</td>
<td></td>
<td>- symmetric stream cipher</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Length of the units of encryption</strong></td>
<td>- determined by block length of the block cipher</td>
<td></td>
<td></td>
<td></td>
<td>+ user-defined</td>
<td></td>
</tr>
<tr>
<td><strong>Error extension</strong></td>
<td>only within the block (assuming the borders of blocks are preserved)</td>
<td>2 blocks (assuming the borders of blocks are preserved)</td>
<td>potentially unlimited</td>
<td>$1 + \left\lceil \frac{b}{r} \right\rceil$ blocks, if error placed rightmost, else possibly one block less</td>
<td>none as long as no bits are lost or added</td>
<td>potentially unlimited</td>
</tr>
<tr>
<td><strong>Qualified also for authentication?</strong></td>
<td>yes, if redundancy within every block</td>
<td>yes, if deterministic block cipher</td>
<td>yes, even concealment in the same pass</td>
<td>yes, if deterministic block cipher</td>
<td>yes, if adequate redundancy</td>
<td>yes, even concealment in the same pass</td>
</tr>
</tbody>
</table>
Collision-resistant hash function using determ. block cipher

*efficient!*

cryptographically strong no, but well analyzed

```
memory for intermediate block n-1

plaintext block n

encryption

last block contains length in bit

initial value is fixed!
(else trivial collisions: intermediate blocks and truncated plaintexts)

birthday paradox after $2^{b/2}$ tests collision
```
An insecure attempt...

$E : K \times \{0,1\}^n \rightarrow \{0,1\}^n$ a block cipher.

Construct cascade, for compression encrypt message blocks:

\[
H_{i+1} = E(m_i, H_i)
\]

What’s wrong with that?

$H_{i+1} = E(m_i, H_i)$

Can you find a collision on this compression function?

$H_{i+1} = E(m', D(m', H_{i+1}))$
The Merkle-Damgard construction

From *short message blocks* to *arbitrarily long messages*...

Given a compression function \( h : \{0,1\}^{2s} \rightarrow \{0,1\}^s \) and

Input \( m \in \{0,1\}^* \) of length \( L \) and \( PB: = 1000...0 \ || \ L \)

Construct \( H \) of \( B = \lceil L/s \rceil \) iterations of \( h \):

If \( h \) is a fixed length CRHF, then \( H \) is an arbitrary length CRHF

**Proof**: either \( M=M' \), or \( H_{B-i}(m[B-i])=H_{B-i}(m'[B-i]) \)

...no collision...collision on \( h \)

29.10.2020 Privacy and Security Folie Nr. 262
practically important: patent exhausted before that of RSA
→ used in PGP from Version 5 on

theoretically important: steganography using public keys

based on difficulty to calculate discrete logarithms

Given a prime number $p$ and $g$ a generator of $\mathbb{Z}_p^*$

$$g^x = h \mod p$$

$x$ is the discrete logarithm of $h$ to basis $g$ modulo $p$:

$$x = \log_g(h) \mod p$$

discrete logarithm assumption
Discrete logarithm assumption

∀ PPA \( \mathcal{DL} \) (probabilistic polynomial algorithm, which tries to calculate discrete logarithms)

∀ polynomials \( Q \)

\exists L \forall l \geq L: \text{(asymptotically holds)}

If \( p \) is a random prime of length \( l \) thereafter \( g \) is chosen randomly within the generators of \( \mathbb{Z}_p^* \)

\( x \) is chosen randomly in \( \mathbb{Z}_p^* \)

and \( g^x = h \mod p \)

\( \mathcal{W}(\mathcal{DL}(p,g,h)=x) \leq \frac{1}{Q(l)} \)

(probability that \( \mathcal{DL} \) really calculates the discrete logarithm, decreases faster than \( \frac{1}{\text{any polynomial}} \))

trustworthy ?? practically as well analyzed as the assumption factoring is hard
Diffie-Hellman key agreement (2)

**Key generation:**
- $x \in \mathbb{Z}_p^*$
- $g^x \mod p$

**Calculating shared key:**
- $(g^y)^x \mod p = g^{xy} = (g^x)^y \mod p$

**Secret area:**

**Domain of trust:**

**Area of attack:**
Diffie-Hellman assumption

Diffie-Hellman (DH) assumption:
Given $p, g, g^x \mod p$ and $g^y \mod p$
Calculating $g^{xy} \mod p$ is difficult.

DH assumption is stronger than the discrete logarithm assumption:

- Able to calculate discrete Logs $\Rightarrow$ DH is broken.
  Calculate from $p, g, g^x \mod p$ and $g^y \mod p$ either $x$ or $y$. Calculate $g^{xy} \mod p$ as the corresponding partner of the DH key agreement.

- Until now it couldn’t be shown:
  Using $p, g, g^x \mod p, g^y \mod p$ and $g^{xy} \mod p$ either $x$ or $y$ can be calculated.
Find a generator in cyclic group $\mathbb{Z}_p^*$

Find a **generator** of a **cyclic group** $\mathbb{Z}_p^*$

Factor $p - 1 = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}$

1. Choose a random element $g$ in $\mathbb{Z}_p^*$

2. For $i$ from 1 to $k$:
   
   $b := g^{\frac{p-1}{p_i}} \mod p$

   If $b = 1$ go to 1.  \( \Leftarrow g \) only generates subgroup!
ElGamal: PKC from Diffie-Hellman

Fix a finite cyclic group $G$ (e.g. $G = (\mathbb{Z}_p)^*$) of order $n$

Fix a generator $g$ in $G$ (i.e. $G = \{1, g, g^2, g^3, \ldots, g^{n-1}\}$)

**Alice**

choose random $a$ in $\{1, \ldots, n\}$

$A = g^a$

**Bob**

choose random $b$ in $\{1, \ldots, n\}$

$B = g^b$

compute $g^{ab} = A^b$

derive symmetric key $k$

encrypt message $m$ with $k$

To decrypt:

compute $g^{ab} = B^a$

derive $k$, and decrypt
The ElGamal System (modern)

G: finite cyclic group of order n

(E_s, D_s): symmetric auth. encryption defined over (K,M,C)

H: \( G^2 \rightarrow K \) a hash function

Construct a pub-key encryption system (Gen, E, D):

Key generation Gen:

- choose random generator \( g \) in \( G \) and random \( a \) in \( Z_n \)
- output \( sk = a \), \( pk = (g, h=g^a) \)

\[
E(pk=(g,h), m): \\
\text{b} \leftarrow Z_n, \text{u} \leftarrow g^b, \text{v} \leftarrow h^b \\
\text{k} \leftarrow H(u,v), \text{c} \leftarrow E_s(k, m) \\
\text{output} \ (u, c)
\]

\[
D(sk=a, (u,c)) : \\
v \leftarrow u^a \\
k \leftarrow H(u,v), \ m \leftarrow D_s(k, c) \\
\text{output} \ m
\]
Digital signature system

Security is asymmetric, too
usually: unconditionally secure for recipient
only cryptographically secure for signer
new: signer is absolutely secure against breaking his signatures
provable only cryptographically secure for recipient

message domain

signature domain

proof of forgery

true

distribution of risks if signature is forged: 1. recipient
2. insurance or system operator
3. signer
Fail-stop signature system

plaintext with signature and test result $x, s(x)$,

"pass" or "fail"

"accepted" or "forged"

random number

key generation

key for testing of signature, publicly known

t

recipient

sign

plaintext with signature $x, s(x)$

key for signing, kept secret

test

"accept" or proof of forgery

generate proof of forgery

verify

plaintext with signature

plaintext with signature

plaintext with signature

plaintext

plaintext with signature

key for signing, kept secret

random number

plaintext with signature

plaintext

plaintext with signature

plaintext with signature

key for testing of signature, publicly known

"pass" or "fail"

"accepted" or "forged"
Undeniable signatures

Interactive protocol for testing the signature

Text with signature and test result $x, s(x)$, “pass” or “fail”

Text with signature $x, s(x)$

Random number

Key for signing, kept secret

Key for testing of signature, publicly known

Test
Signature system for blindly providing of signatures

RSA

\[ p \cdot q = n \]

Key generation

Random number

\[ x \cdot z^t \]

Blind

Text

\[ x \]

Random number

\[ z' \]

Blinded text

\[ (x \cdot z^t)^s = \]

Sign

\[ x^s \cdot z' \]

Unblind and test

\[ z'(x), s(z'(x)) \]

Text with signature and test result

\[ x, s(x), x^s \]

"pass" or "fail"
Threshold scheme for secret sharing (1)

**Threshold scheme:**

- Secret $S$
- $n$ parts
- $k$ parts: efficient reconstruction of $S$
- $k-1$ parts: no information about $S$

**Implementation: polynomial interpolation (Shamir, 1979)**

**Decomposition of the secret:**

Let secret $S$ be an element of $\mathbb{Z}_p$, $p$ being a prime number.

Polynomial $q(x)$ of degree $k-1$:

Choose $a_1, a_2, \ldots, a_{k-1}$ randomly in $\mathbb{Z}_p$

$q(x) := S + a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}$

$n$ parts $(i, q(i))$ with $1 \leq i \leq n$, where $n < p$. 
Threshold scheme (2)

Reconstruction of the secret:

$k$ parts $(x_j, q(x_j)) \ (j = 1 \ldots k)$:

\[
q(x) = \sum_{j=1}^{k} q(x_j) \prod_{m=1, m \neq j}^{k} \frac{(x - x_m)}{(x_j - x_m)} \mod p
\]

The secret $S$ is $q(0)$.

Sketch of proof:
1. $k-1$ parts $(j, q(j))$ deliver no information about $S$, because for each value of $S$ there is still exactly one polynomial of degree $k-1$.
2. correct degree $k-1$; delivers for any argument $x_j$ the value $q(x_j)$ (because product delivers on insertion of $x_j$ for $x$ the value 1 and on insertion of all other $x_i$ for $x$ the value 0).
Threshold scheme (3)

Polynomial interpolation is Homomorphism w.r.t. addition

Addition of the parts ⇒ Addition of the secrets

Share refreshing

1.) Choose random polynomial \( q' \) for \( S' = 0 \)
2.) Distribute the \( n \) parts \((i, q'(i))\)
3.) Everyone adds his “new” part to his “old” part
   → “new” random polynomial \( q + q' \) with “old” secret \( S \)

• Repeat this, so that anyone chooses the random polynomial once
• Use verifiable secret sharing, so that anyone can test that polynomials are generated correctly.
• …for thesis, project work, paid jobs etc.

• When:
  – 7.2. – 13:00

• Where:
  – APB 3105