Course Logistics and Introduction

Cryptography, Autumn 2021

Lecturers: J. Daemen, B. Mennink

September 7, 2021

Institute for Computing and Information Sciences
Radboud University

Outline

Course basic info

Administrative details

Crypto basics refresh

Course basic info

What will you learn

- Cryptographic primitives, schemes and protocols used in the real world
  - definition of security goals
  - design rationale: how are the goals achieved
- Questions we aim at answering
  - how are cryptographic schemes constructed and why?
  - what does it mean for a scheme to be secure?
  - how do we quantify security strength?
- Basics of underlying mathematics:
  - modular arithmetic and elementary number theory
  - finite groups and fields

Pre-requisites

- Basic principles of cryptographic services and protocols
  - as taught in bachelor course Security (NWI-IPC021)
  - this course takes off where Security stopped
- Basics of linear algebra, combinatorics and probability theory
  - as taught in following bachelor courses
  - Mathematical Structures (NWI-IPC020)
  - Combinatorics (NWI-IBC016)
  - Matrix Calculation (NWI-IPC017)

Intro to Crypto is a pre-requisite itself for the RU cybersecurity master

Administrative details

What, When, How

- Weekly 4 hours: hybrid lectures and physical tutorials
  - we expect you to follow the lectures
- Lectures on Tuesdays 13:30-15:15 in LIN 3
  - cover new concepts and theory
  - 75 students can attend, registration is mandatory
  - remaining students can follow via a livestream
  - recordings will be made available in Brightspace
  - **lecture on Thursday September 9 online**
- Tutorials on Thursdays 10:30-12:15 and 13:30-15:15
  - practice course material by working on assignments
  - location in Huygens: see course manual or persoonlijkrooster
  - sign-up through Brightspace later this week

Who

We're all in Mercator I (room number, see below)

- Lecturers:
  - prof. Joan Daemen, 3.19 (course coordinator)
  - Bart Mennink, 3.15
- Teaching assistants
  - Bobby Subroto, 3.03
  - Jan Schoone, 3.11b
- email addresses: firstname.lastname@ru.nl

Grading

The final grade consists of:

- 10% homework (in pairs, weekly homework assignments)
- 20% mid-term test (individually)
- 70% final exam (individually)
  - exam questions aligned with homework problems
  - to pass, you must score at least 50% on the final exam

In case of resit:

- 10% homework (original grades)
- 90% resit exam (individually)

Exams are on-campus and written

Lecture and tutorial schedule

Week	Tuesday		Thursday	
36	September 7	Lecture 1	September 9	Lecture 2
37	September 14	Lecture 3	September 16	Tutorial 1
38	September 21	Lecture 4	September 23	Tutorial 2
39	September 28	Lecture 5	September 30	Tutorial 3
40	October 5	Lecture 6	October 7	Tutorial 4
41	October 12	Lecture 7	October 14	Tutorial 5
42	October 19	Q & A	October 21	Tutorial 6
43-44	midterm exam, November 1			
45	November 9	Lecture 8	November 11	Tutorial 7
46	November 16	Lecture 9	November 18	Tutorial 8
47	November 23	Lecture 10	November 25	Tutorial 9
48	November 30	Lecture 11	December 2	Tutorial 10
49	December 7	Lecture 12	December 9	Tutorial 11
50	December 14	Lecture 13	December 16	Tutorial 12
51	December 21	Lecture 14	December 23	Tutorial 13
	final exam, January 17			
	resit, *to be scheduled*			

Homework schedule

- Assignment in Brightspace: Wednesday 10:00, week n
- You can ask advice in tutorials of week n and $n + 1$
- Hand-in deadline: Monday 17:00 of week $n + 2$
- Grade in Brightspace: we aim for the Monday in week $n + 3$
- First assignment is online Wednesday 15 September
- From then on one in each lecture week

General rule: too late means score 0, no exceptions

Resources, all available on Brightspace

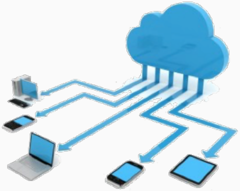
- Course manual: schedules, rules and practical information
- Slides
  - **are the reference**
  - are available in Brightspace
  - may be updated after the lecture
- Lecture recordings
  - allows you to re-visit lectures
  - not meant as substitute for attending the lectures (physical or via livestream)
- Lecture notes
  - intended to complement the slides for studying
  - contains informative parts that are not exam material
  - we started them 3 years ago, still work-in-progress
  - all feedback welcome (to main author Jan Schoone)
  - updates will be made available in Brightspace

Crypto basics refresh

Cryptography is everywhere nowadays








What do we want to protect?

The classical security services:

- Confidentiality AKA data privacy: the assurance that data cannot be viewed by an unauthorised party
- Data integrity: the assurance that data has not been modified in an unauthorised manner
- Data origin authentication: the assurance that a given entity was the *original source* of received data
- Entity authentication: the assurance that a given entity is who she/he/it claims to be
- Non-repudiation: the assurance that a person cannot deny a previous commitment or action

Basic Data Confidentiality

- To protect:
  - people's privacy
  - company assets
  - enforcing business model: no pay, no content
  - PIN, password, cryptographic keys
- Data confidentiality
  - only authorised entities get access to the data
  - cryptographic operation to enable this: encryption
  - encryption and decryption share secret key
- Requires sender and receiver to establish shared secret key

Advanced Confidentiality: anonymity, unlinkability etc.

Example: Protection against traffic analysis

- threats due to frequency and statistics of communication
- exploiting so-called metadata
- countermeasure: hiding communication between entities
- cannot be provided by cryptography alone but additionally:
  - dummy messages
  - random-length padding
  - *mixnets*, ...

Non-repuditation

- Previous commitment or action cannot be denied
  - in front of an arbiter or judge
  - cryptographic material serves as evidence
- Concept of legal or regulatory type
- Assuring it requires more than crypto: system-level approach
  - lawyer may exploit any security hole
  - in case of trial typically experts are called in
- Often realized by contract, law or directive rather than cryptography

Cryptographic function for authentication: MAC function

- It is widely believed that encryption protects plaintext integrity
  - "if decryption gives valid plaintext, it was not altered"
  - this is in general not true
  - encryption does not provide integrity, so no authentication
- Message authentication code (MAC)
  - cryptographic checksum (tag) over message, challenge . . .
  - lightweight cryptographic operation
- Requires prover and verifier to establish shared secret key

Cryptographic function for authentication: signature

- (Digital) Signature: cryptographic counterpart of real-life signing
  - cryptographic checksum over message, challenge ...
  - rather heavyweight cryptographic operation
  - signer uses a private key it does not share with anyone
  - verifier only needs public key of the signer
- Requires verifier to authenticate signer's ownership of public key
- Reasons to use signature rather than MAC
  1. auth. of broadcast messages, e.g., software updates
  2. signature as evidence for a judge/arbiter (non-repudiation)
  3. if verifier not known in advance, e.g., travel passport

Freshness and resistance against replay attacks

- Freshness:
  - entity is there now
  - received message was written recently
  - mechanism: include unpredictable challenge in MAC/signature computation
  - unpredictable challenge must come from verifier
- Protection against replay:
  - authenticated message was not just a copy of an earlier one
  - mechanism: include nonce in MAC/signature computation
  - verifier must check uniqueness of nonce

Establishment of a secret key

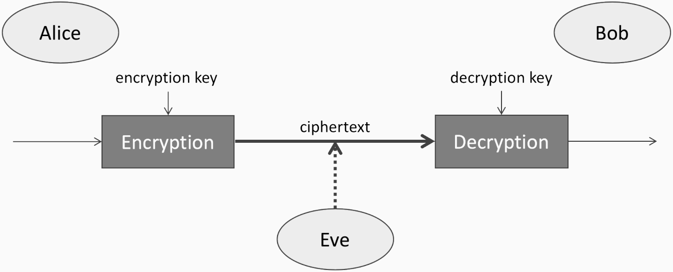
- What?
  - dedicated cryptographic operation
  - different from encryption, MAC and signature
  - establishment of shared secret between Alice and Bob
  - shared secret to be used as key to protect data
- Goal: confidentially establish a key by exchange of public information
- Can achieve forward secrecy
  - "compromise of endpoint (PC, phone, ...) does not jeopardize confidentiality of old communications"
  - requires private keys or secrets used for key establishment to be deleted from the endpoint after usage

Secure channel

- Cryptographically secured link between two entities
- Data confidentiality and authentication
- Session-level authentication
    - insertion, removal, shuffling of messages
- Can be one-directional or full-duplex
- Can be online or store-and-forward
- Can require freshness or just protection against replay
- Examples: SSH, TLS, WhatsApp, Signal, WPA, Skype...

Adversary (or attacker) model

The classical encryption use case:



- Alice: sender
- Bob: receiver
- Eve (eavesdropper): adversary

Modern use cases are more complex and Eve may have more access:

Adversary Model

Specification of what we assume an adversary can do and access

How are cryptographic schemes built?

- Lego approach:
  - modern cryptographic schemes are modular
  - atomic building blocks: primitives
  - using constructions or modes
- Example: AES-CBC
  - is an encryption scheme supporting arbitrary-length messages
  - primitive: block cipher AES
  - mode: CBC, specifying how to apply the block cipher
- Protocols
  - implies interaction between different entities
  - makes use of cryptographic schemes
- Security goals must be clear and well-defined
  - apply to primitives, schemes and protocols
  - quantitative: *security strength*
  - always with respect to an adversary model (sometimes implicit)
  - many systems are complex and/or wrong due to ill-defined goals

Analysing security of a cryptographic scheme/protocol

- Understand security goals that a scheme/protocol should meet

(1) Define the adversary model
   - what is the adversary's goal?
   - what is the adversary's power?
   - this defines the requirements the solution must meet
   - verify that the adversary model fits the application

(2) Express a solution (protocol or scheme) that addresses the requirements
   - use constructions and modes that allow to reduce the requirements on the construction to that of primitives
   - show that an adversary cannot break the scheme without breaking the underlying primitive
   - use primitives that are believed to satisfy those requirements

Provably secure primitives

- ... exist but are hardly ever practical
- It means one can prove security strength is above some (decent) level
- Still some security aspects of it may be provable
- Provable constructions
  - secure if ideal underlying primitives
  - remaining problem: build a primitive that behaves ideally
- Security proofs by reduction
  - breaking implies solving famous hard problem (e.g., factoring)
  - credibility depends on understanding of hard problem
  - typical for public key cryptography
  - problem: famous problems are often not so well understood

The basis for trust in cryptographic primitives

- The (open) cryptologic activity:
  - cryptographic primitives are published
  - ...and (academically) attacked by cryptanalysts
  - ...and corrected/improved,
  - ...and attacked again, etc.
  - by researchers for prestige/career
- This leads to
  - better understanding
  - ever stronger cryptographic primitives
- Trust in cryptographic primitive depends on
  - reputation of designers
  - perceived simplicity
  - perceived amount of analytic effort invested in it
  - reputation of cryptanalysts

Security claim

- Lack of security proof leaves following questions unanswered:
  - what kind of security does a particular primitive offer?
  - when does a demonstrated weakness constitute an attack?
- This is addressed by a *security claim*

Security claim

Precise statement on expected security of a cryptographic primitive

- Serves as challenge for cryptanalysts
  - break: attack performing better than the claim
- ...and security specification for user
  - ...as long as it is not broken

Often claims are missing but implied by size parameters such as key length, tag length, digest length ...

What does a typical security claim look like

Not: this scheme is impossible to break, but rather

- Success probability of *breaking the primitive* by an adversary with following well-defined resources:
  - N: amount of computation, in some well-specified unit
  - M: amount of input/output computed with the secret key
  - possibly limitations on memory usage, ...
- ... is upper bound by $\epsilon(N, M)$
- ϵ is typically very small as a function of N and M

Example of a claim:

PRP security of AES-128 (see later)

Distinguishing AES with 128-bit secret key from a random permutation has success probability $\leq 2^{-128}N$

Often shortened to: *AES-128 offers 128 bits of security (strength)*

Security strength

- Quantifies the expected/claimed security of a primitive, in *bits*
- Historically, security strength s bits means:
  - breaking primitive requiring resources $M + N = 2^s$, and/or
  - attack with minimal resources having success prob. $p = 2^{-s}$

Security strength (modern definition)

A cryptographic scheme offers security strength s if there are no attacks with $(M + N)/p < 2^s$ with N and M the adversary's resources and p the success probability

Current view (see e.g. `www.keylength.com`)

- 56 bits: not secure
- 80 bits: lightweight
- 96 bits: solid
- 128 bits: secure for the foreseeable future
- 256 bits: for the clueless

Data versus computational complexity

- There is an important difference between the two types of resources available to the adversary
- N: amount of computation. Has different names
  - *computational complexity*: for obvious reasons
  - *time complexity*: as it typically spends time on a CPU
  - *offline complexity*: offline from attacked instance
  - the only limit to N is the wealth of the attacker
- M: amount of input/output computed with the secret key. Names:
  - *data complexity*: data as obtained from the attacked instance
  - *online complexity*: online with attacked instance
  - can be limited by designing protocols in smart way
- Security strength often makes abstraction of distinction between these two very different complexities
- More fine-grained statements about security strength express $s = \log_2 N/p$ given certain limitations on M