Radbound University

Course Logistics and Introduction

Cryptography, Spring 2020

L. Batina, J. Daemen

February 5, 2020

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 cryptographic schemes are constructed and why
  - what does it mean for a scheme to be secure
- Basics of underlying mathematics:
  - modular arithmetic and elementary number theory
  - finite groups and fields

Pre-requisites

- Computer Security Course
  - that course discusses protocols using cryptographic schemes
  - Intro to Crypto takes off where Security stopped
- Basic understanding of:
  - combinatorics
  - linear algebra
  - probability theory

What this course does not cover

This is intro to crypto, not more, not less

More specialized topics are treated in other courses, e.g.,

- Securely implementing crypto in Cryptographic engineering
- Embedded systems security in Hardware security
- Firewalls, network sniffing and traffic analysis in Network security
- UNIX security, malware detection in OS security

Administrative details

What, Who, When, Where

- Weekly 4 hours: lectures and tutorials
  - we expect you to be present and active
- Course coordinator:
  - prof. Lejla Batina, Mercator 1, 3.10, lejla@cs.ru.nl
- Lectures: cover new concepts and theory
  - on Wednesdays 15:30-17:15 in Huygens Auditorium 307
  - by prof. Joan Daemen, Mercator 1, 3.19, joan@cs.ru.nl
- Tutorials: practice course material by working on assignments
  - on Thursdays 15:30-17:15, in Huygens Auditorium 304
  - Jan Schoone, Mercator 1, 3.20, j.schoone@cs.ru.nl
  - Marloes Venema, Mercator 1, 3.17, m.venema@cs.ru.nl
  - Jonathan Fuchs, Mercator 1, 3.17, j.fuchs@cs.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
  - for passing you must score at least 50 % on the final exam

In case of resit:

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

Note: this is incorrect in course guide due to an administrative mistake and will be corrected

Lecture and tutorial schedule

	Wednesday		Thursday	
week 6	February 5	Lecture 1	February 6	Lecture 2
week 7	February 12	Lecture 3	February 13	Tutorial 1
week 8	February 19	Lecture 4	February 20	Tutorial 2
week 9	February 26	Lecture 5	February 27	Tutorial 3
week 10	March 4	Lecture 6	March 5	Tutorial 4
week 11	March 11	Lecture 7	March 12	Tutorial 5
week 12	March 18	Lecture 8	March 19	Tutorial 6
week 13-15		intermediate exam, April 2		
week 16	April 15	Lecture 9	April 16	Tutorial 7
week 17	April 22	Lecture 10	April 23	Tutorial 8
week 18		-		-
week 19	May 6	Lecture 11	May 7	Tutorial 9
week 20	May 13	Lecture 12	May 14	Tutorial 10
week 21	May 20	Lecture 13	-	
week 22	May 27	Lecture 14	May 28	Tutorial 11
week 23	June 3	Lecture 15	June 4	Tutorial 12
week 24	June 10	Lecture 16	June 11	Tutorial 13
week 25		-		-
week 26		final exam, June 25		

Homework schedule

- Assignment in Brightspace: Thursdays 10:00, week n
- You can ask advice in tutorials of week n and $n+1$
- Hand-in deadline: Friday 12:00 of week $n+1$
- Grade in Brightspace: we try before tutorial week $n+2$
- First assignment is on Thursday 6 February
- From then on one in each lecture week

General rule: too late means score 0, no exceptions

Resources

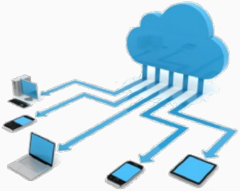
- Slides
  - **are the reference**
  - we'll try to have them in Brightspace before the lecture
- Lecture notes
  - intended to complement the slides for studying
  - contains also informative parts that are not exam material
  - we started them last year, still room for improvement
  - all feedback welcome
  - updates will be made available in Brightspace

Crypto basics refresh

Cryptography is everywhere nowadays








Cryptography is about communication in the presence of adversaries
Ron Rivest

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 Confidentiality

- To protect:
  - people's privacy
  - company assets
  - enforcing business: 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
  - remaining problem: establishment of 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-repudiation

- 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
  - experts are typically called in case of trial
- Often just by contract, law or directive

Cryptographic functions for authentication: MAC and signature

- Message authentication code (MAC)
  - cryptographic checksum (tag) over message, challenge ...
  - lightweight cryptographic operation
  - requires generator and verifier to share secret key
  - remaining problem: establishment of secret key
- (Electronic) Signature
  - 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 sender
  - remaining problem: authentication of link between public key and owner of corresponding private key
- Reasons to use signature:
  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

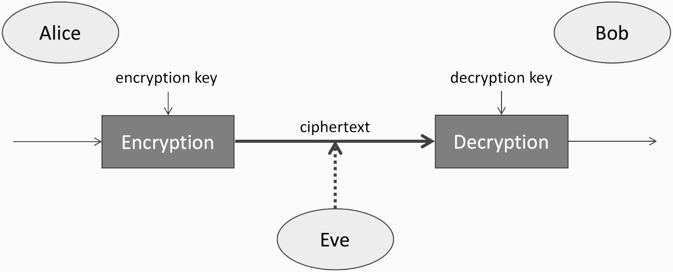
- Dedicated cryptographic operation
- Different from encryption, MAC and signature
- Establishment of shared secret between Alice and Bob
- Shared secret to be used as key to protect data
- Goal: establish a key while ensuring its confidentiality while only exchanging public information
- Can achieve forward secrecy
  - forward secrecy means that 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
- E.g., encryption scheme supporting arbitrary-length messages
  - primitive: block cipher
  - mode: CBC mode with padding of last block and nonce
- Protocols
  - implies interaction between different entities
  - make use of cryptographic schemes
- Security goals must be clear and well-defined
  - apply to primitives, schemes and protocols
  - 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

- . . . exist but are hardly ever practical
- 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 quite obscure

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 $\epsilon = f(N, M)$
- 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