Engineering Secure Software Systems

November 3, 2020: Intro

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

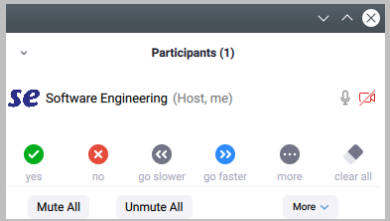
You hear me?

1, 2, 3, …

Echo fail?

- Chat (I'll be half-watching)
- "Raise Hand"
- use your voice
- EMail

fast feedback



Introduction: Me

Bio Henning Schnoor

2004 diploma mathematics / computer science in Hannover

2004-7 PhD in Hannover advised by Prof. Heribert Vollmer
Algebraic Techniques for Satisfiability Problems

2007-8 Postdoc in Rochester, NY, USA

2008-15 Habilitation in Kiel, advised by Prof. Thomas Wilke
Knowledge-based, Strategic, and Temporal Security Properties

s. 2016 Privatdozent, AG Software Engineering, Prof. Wilhelm Hasselbring

Research interests

- formal methods for IT security
- metrics for software quality
- complexity theory
- logic
- computational social choice

Introduction: You

study program

- bachelor computer science
- master computer science
- Wirtschaftsinformatik
- mathematics
- physics
- others?

Admin

Overview

Admin

Accounts, Materials

Distance Learning in ESSS 20/21

Administrative

time and place

lecture tuesday, 12:15-13:45

exercise thursday, 9:00-9:45

url `https://uni-kiel.zoom.us/j/84262484452?pwd=bVNkeDJiWXk1c0ZpaE5sWnBOeUhDUT09`

passcode 699272

exercise and exam

- as usual: work in pairs
- exam: depending on number of students after $\approx$ 6 weeks
  - oral exam
  - oral exam for active students
  - written exam
- no bonus points for exam grade
- dates depend on exam period (Prüfungszeitraum)

ESSS Exercise Class

exercise class goals

- apply concepts and formalisms from lecture yourself
- time for details and clarifications
- I hope: discussions and questions

exercise content: tasks

- tasks presented in lecture, also published as exercise sheet
- review questions from lecture
- work on tasks before discussion in exercise class
- hand-in/feedback using git repositories
- not required to solve tasks for good grade
- discussion in exercise class after hand-in
- first sheet: released today

Materials: Lecture and Exercise

materials: my repository

- slides, lecture notes, exercise sheets, modeling scripts …
- *notes*: additional material for “marked” slides
  - background material, formal proofs, discussion
  - review questions

further reading

- no textbook
- references: original research papers

materials repo clone command

```
git clone https://hs:e3eLXb8-iLb3hoG7LRGo@git.informatik.uni-kiel.de/hs/esss-ws2021-common.git
```

exercise solutions: your repository

- form groups of 2 students to work together
- create GitLab project to hand-in exercises

Exercise

Task (exercise git project)

Create a git project together with your exercise partner at `https://git.informatik.uni-kiel.de` using the naming scheme `LL-SEM-Lastname1-Lastname2` and add Henning Schnoor (username hs) as a **Maintainer** to your project. Usually, you should have an account from your Bachelor's studies. If you did not obtain your Bachelor in Kiel or do not have such an account for some other reason, see `http://www.inf.uni-kiel.de/de/service/technik-service/accounts` for details on how to obtain such an account. In the project name, `LL` is an abbrevation for the lecture, (e.g., **SEPVS** for Software Engineering für Parallele und Verteilte Systeme or **ESSS** for Engineering Secure Software Systems), **SEM** is an abbrevation for the semester, like **WS20** for Winter 2020/2021. Lastname1 and Lastname2 are the last names of the two students in the working group. Write an email to Henning Schnoor with the URL of the repository (it suffices for one student in each group to write this mail).

Handing in of exercises and feedback to your tasks will use this git account. For non-programming exercises, answers must be submitted in one of the formats pdf, markdown, or plain text.

Overview

Admin

Accounts, Materials

Distance Learning in ESSS 20/21

Distance Learning: My Experience

on-campus

- direct interaction with students
- judge your reactions, see what works and what does not
- make corresponding adjustments
- ideally: interested, but critical audience

two possibilities

1. my teaching is perfect, nothing to adjust
2. something's missing …

distance learning

- very limited interaction beyond multiple choice polls, chat questions
- full Zoom participants lists
- lectures go exactly as planned
- very uncritical audience: my wall!

Distance Learning: IfI Experience

experience IfI

- very little interaction
- feedback mostly when using polls / surveys in class, some questions in chat
- students are essentially names in Zoom participant lists
- Zoom-teaching almost the same as video recording

summary

- teaching in summer 2020: essentially video tutorials plus homework, exam grading
- works for basic programming courses, but not here

Adjustments

then and now

- summer 2020: used on-campus methods in remote setting
- now: we're better prepared, try "real" remote teaching methods

usage of class time

- goal: less prepared presentation, more interaction
- your participation:
  - questions
  - discussion of review questions
  - presentation of exercise solutions
- consequence: no video recordings of "live lectures"

Distance Learning in ESSS 20/21: Half a Plan!

weeks ≈ 1-3: introduction, motivation, formal model introduction

traditional lecture

- “live” presentation here in class
- discussion of review questions in (exercise) class

weeks ≈ 4-6: formal model application, main (un)decidability results

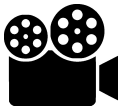
flipped classroom

- I provide materials (videos, lecture slides, lecture notes) beforehand, including “schedule”
- use “Wilke model” in class: meet in smaller groups to discuss questions

after ≈ week 6

review what worked with your feedback

Video Lectures



motivation

- non-interactive presentations do not need to be “live”
- technical content like proofs better presented in videos than on slides

video content

- slides with voiceover
- hand-written proofs on paper/whiteboard
- voice-over

alternative to videos

- slides from videos appear in lecture slides (gray background for distinction)
- detailed proofs contained in lecture notes

Workload?

Workload?

your perspective

$=$ regular lecture meetings (some only partial with split groups)

$=$ work on exercise tasks

$+$ work on review questions

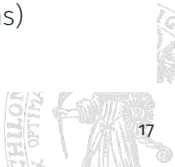
$+$ watch video lectures

issue

- more work compared with traditional course?
- apples/oranges comparison: in on-campus iteration, you also study material on your own!
- distance learning can't just be "classroom relocated to Zoom"
- my hypothesis/goal: workload for **active participants** similar as in on-campus iteration

measure to limit workload for you

- we cover the same topics as in last iteration of the course (also 13 lecture sessions)
- lecture progress: comparable to last time



How can we make this work?

cameras

- experience: people are more focused in online meetings when cameras are on
- but: we're many people
- suggestion: turn your camera on at least during "discussion parts" of the lecture

Zoom

- use real names
- use "raise hand" function, talk if I don't "see you"
- questions in chat: I try to see and answer them all, but usually with delay

general

your feedback is essential!

Distance Learning: Key Points

none of this works without your participation!
also on-campus: active students are more successful in this course

we're still learning!
let's experiment and try some stuff!

"I'm just here for a quick look"
if you just want to watch/listen to get a superficial overview of the topic: fine too!

Overview and Motivation

Overview

relevance

- topic gets (relatively) much media attention
- usual reporting: current breaches and attacks
- high-profile examples
  - RAMBleed, Meltdown, Spectre, Heartbleed
  - WPA/2 attack KRACK
  - Loss of Credit Card / Hotel Customer Data
  - malware (e.g., crypto miners in audio files)
  - concrete security issues in specific software
  - …

WPA/2 KRACK

summary

- Wi-Fi Protected Access 2 (IEEE 802.11i): WLAN security
- designed to ensure privacy over WLAN
- widespread deployment

KRACK (October 2017)

- Mathy Vanhoef and Frank Piessens. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. 2017 (presentation based on, images taken from this paper)
- crucial bug in the protocol
- allows attacker to read all messages
- crucial for this lecture: conceptual issue, not only bug in concrete implementation

Attack Outline

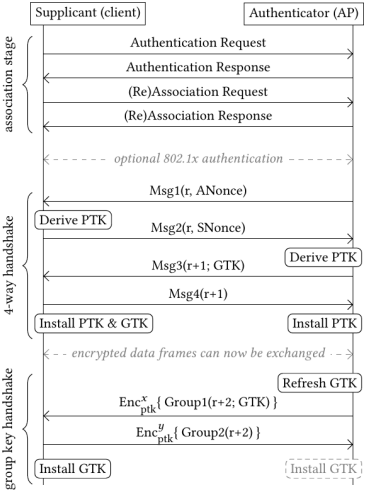
vulnerability

- WPA/2 uses a sub-protocol for encrypted communication
  - CCMP (Counter-Mode/CBC-MAC Protocol), based on AES
  - secure as long as “initialization vectors” (some random numbers) are not reused
- key to attack: force protocol to re-use initialization vectors
- then CCMP does not guarantee any security anymore

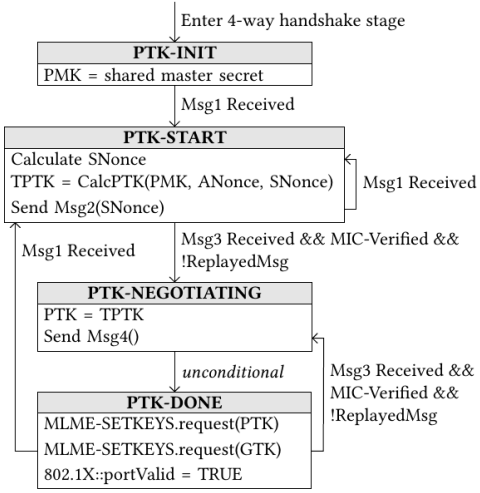
question

how does an attacker force re-use of initialization vectors?

Handshake Specification I



Handshake Specification II



Bug in Protocol

approach

- protocol allows retransmission/acceptance of message from handshake
- needed in case of message loss
  - can occur with unreliable network
  - or be forced by attacker
- effect: CCMP used with same “initialization vector”
- completely breaks WPA/2

worst case

- Linux systems (including Android): key reinstallation leads to “zero key”
- trivial to “crack”

Affected Systems

Implementation	Re. Msg3	Pt. EAPOL	Quick Pt.	Quick Ct.	4-way	Group
OS X 10.9.5	✓	✗	✗	✓	✓	✓
macOS Sierra 10.12	✓	✗	✗	✓	✓	✓
iOS 10.3.1 [c]	✗	N/A	N/A	N/A	✗	✓
`wpa_supplicant` v2.3	✓	✓	✓	✓	✓	✓
`wpa_supplicant` v2.4-5	✓	✓	✓	✓[a]	✓[a]	✓
`wpa_supplicant` v2.6	✓	✓	✓	✓[b]	✓[b]	✓
Android 6.0.1	✓	✗	✓	✓[a]	✓[a]	✓
OpenBSD 6.1 (rum)	✓	✗	✗	✗	✗	✓
OpenBSD 6.1 (iwn)	✓	✗	✗	✓	✓	✓
Windows 7 [c]	✗	N/A	N/A	N/A	✗	✓
Windows 10 [c]	✗	N/A	N/A	N/A	✗	✓
MediaTek	✓	✓	✓	✓	✓	✓

[a] Due to a bug, an all-zero TK will be installed, see Section 6.3.
[b] Only the group key is reinstalled in the 4-way handshake.
[c] Certain tests are irrelevant (not applicable) because the implementation does not accept retransmissions of message 3.

But there was a Proof?

The 4-way handshake was mathematically proven as secure. How is your attack possible?

Our attacks do not violate the security properties proven in formal analysis of the 4-way handshake. In particular, these proofs state that the negotiated encryption key remains private, and that the identity of both the client and Access Point (AP) is confirmed. **Our attacks do not leak the encryption key.** Additionally, although normal data frames can be forged if TKIP or GCMP is used, an **attacker cannot forge handshake messages and hence cannot impersonate the client or AP during handshakes**. Therefore, the properties that were proven in formal analysis of the 4-way handshake remain true. However, the problem is that the proofs do not model key installation. Put differently, **the formal models did not define when a negotiated key should be installed**. In practice, this means the same key can be installed multiple times, thereby resetting nonces and replay counters used by the encryption protocol (e.g. by WPA-TKIP or AES-CCMP).

Lessons Learned

take-aways

- bugs appear in standard, widely used protocols
- composition of security protocols not trivial
  - approaches see [Can01], [BPW04], [Küs06], [Lin17]
- formal methods
  - methods only verify specified (security) properties
  - specifying security properties itself is highly nontrivial
  - approaches see [Sch12] (and references therein), [NSC18]
- definitions of security critical
  - no “single security definition”
  - consequence: security specification should be part of an analysis algorithm’s input

Engineering Secure Software Systems

Wikipedia

Engineering is the application of **mathematics**, as well as **scientific**, economic, social, and **practical** knowledge, to invent, innovate, **design**, build, maintain, **research**, and improve structures, machines, **tools**, **systems**, components, materials, processes, solutions, and organizations.

lecture focus: topics that

- are parts of a unified theory
- tell us how to **build** secure (parts of) systems and **tools** to analyse them
- analyse security of one **aspect** (level of abstraction)

Engineering: We love Tools!

software engineering

- IDEs
  - refactoring capabilities
  - type-aware code completion
  - …
- static analysis
  - checkstyle
  - PMD
  - …
- dynamic analysis
  - monitoring
  - trace analysis
- …

security engineering

?

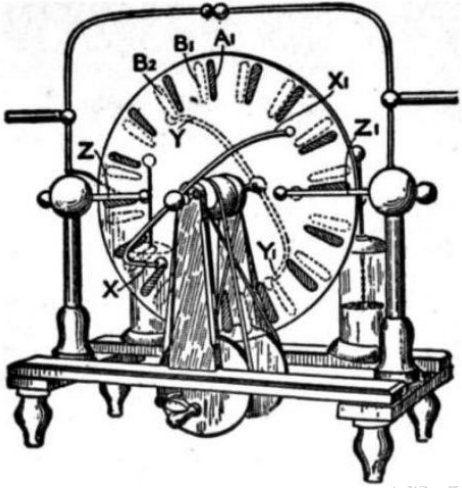
this lecture

- what do we **want** tools to do?
- what can—and cannot—be done?

theory lecture!

- study theory behind tools
- algorithms, hardness and impossibility results
- practical aspect: ProVerif

Ideal Situation



push-button tool

input system, security criterion

output SECURE or INSECURE

this lecture

how close can we come (for two areas)?

first question

what does “secure” mean? need formal model!

Key Areas in Lecture

key topics

1. cryptographic protocols
   - **formal model**
   - definitions of security
   - **automatic analysis**
   - real examples: voting protocols, (Bitcoin,) …
2. information-flow security
   - **formal model**
   - transitive and intransitive policies
   - **automatic analysis**
   - real example: Meltdown
3. language-based security
   - if we get around to it!

theory lecture

security: success story for formal methods

- abstract systems view
  1. Alice-and-Bob protocols
  2. system as state diagram
  3. “toy examples” for languages
- results
  - security proofs (to a point)
  - algorithms, (im-)possibility results
  - complexity results
- techniques
  - **formal models and proofs**

Caveat: Theory Lecture

theory lecture
- formal models
- formal definitions
- formal results and **proofs**.

compromise approach
- theory in detail
- examples: a bit of handwaving
- additional details are in the notes!

motivation from practice
- need theory that can express real systems
- want to express real systems in the theory
- formal models of real systems tend to be large

issues
- there is no “correct” level of formality
  - lecture feedback goes both ways, I promise to err on both sides!
- if you want more formal details / more intuitive examples, let me know!

Caveat: Theory Lecture

prerequisites

- foundations of computer science theory (TGI)
  - complexity: nondeterministic algorithms, reductions, NP-completeness
  - decidability: reductions, undecidability
- logic for computer science (LogInf)
  - terms and signatures
  - Horn clauses, first-order logic
- data structures and algorithms (ADS)
  - data structures, trees, (directed acyclic) graphs

Overview

what are crypto protocols?

- application of crypto primitives to provide “secure” services
- examples: Diffie-Hellman, Internet Key Exchange, TLS, …

Needham-Schroeder protocol

$$
\begin{aligned}
A &\to B \quad \mathrm{enc}^{\mathrm{a}}_{k_B}(A, N_A) \\
B &\to A \quad \mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, N_B) \\
A &\to B \quad \mathrm{enc}^{\mathrm{a}}_{k_B}(N_B)
\end{aligned}
$$

Analysis of Protocols

goal: automatic analysis

analyse security properties of protocols (semi-)automatically

feasibility

- protocols are “small,” so complete analysis possible
- successful application of formal methods
  - model checking
  - (interactive) theorem proving
  - type systems

difficulty

protocols “small,” but attacker strategies unrestricted

question

how far can we go?

Overview

Information Flow Security

motivation

- components with different security levels on one system
- `top secret` $>$ `classified` $>$ `public`
- requirement: no information about “higher level” data can be derived from “lower level” data

questions

- what does “secure” mean?
- how can we check / guarantee security?

points of view

local system as finite automaton

global architecture view

Declassification

requirement

`top secret` data may not influence `public` data

practice

need exceptions

- information flow needed in some cases
- which ones?
- how do we make sure there are no other exceptions?

Overview

Learning Goals

protocols

- specify protocols and security properties
- evaluate security
- automatic analysis and its limits
- best practice for protocol design
- transfer of abstract results to real protocols
- model and analyse protocols with ProVerif

information flow

- know security notions and relationships
- choose matching security notion for application scenario
- evaluate systems, know basic algorithms

language based security

- formal and “practical” security

Lecture Plan (based on last year)

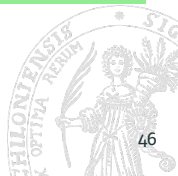
categories

- introduction
- examples
- formal model
- algorithms and proofs
- tool application

lecture 1 — November 3, 2020 intro, overview, crypto primitives	lecture 2 — November 10, 2020 examples, protocol model, adversary capabilities
lecture 3 — November 17, 2020 protocol model, attack definition	lecture 4 — November 24, 2020 RT algorithm and proof
lecture 5 — December 1, 2020 RT algorithm and proof	lecture 6 — December 8, 2020 RT proof II, negative results
lecture 7 — December 15, 2020 logic modeling, ProVerif basics, equational theories	lecture 8 — January 5, 2021 ProVerif: secrecy and beyond
lecture 9 — January 12, 2021 ProVerif: events and incompleteness	lecture 10 — January 19, 2021 Voting Protocols
lecture 11 — January 26, 2021 infoflow basics, transitive infoflow	lecture 12 — February 2, 2021 transitive and intransitive infoflow
lecture 13 — February 9, 2021 unwindings	

skipped in winter 19/20

abstractions, BitCoin



Part I: Crypto Protocols

Overview

Part I: Crypto Protocols

Foundations

Cryptography

What are Cryptographic Protocols?

protocols

- fix steps of communication
- sequence of messages
- examples

TCP	“low-level” communication
HTTP	website delivery
SMTP	email transport

assumption for application

- parties are honest
- reliable channels (TCP protects against non-malicious errors)

internet: both assumptions unrealistic!

Well-Known Crypto Protocols

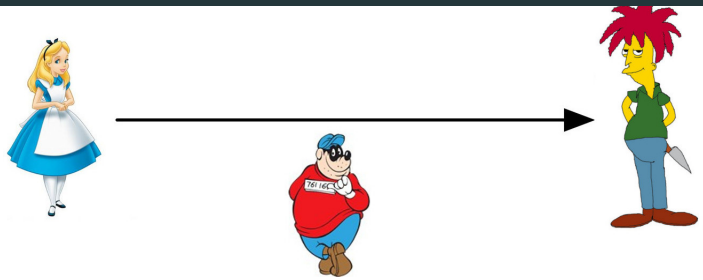
examples

SSL/TLS	encryption, authentication of web communication
IPSec	virtual private networks
IKE	internet key exchange
SSH	secure (remote) shell
BitCoin	digital currency

security

- Are these protocols “secure”?
- What does “secure” even mean?
  recall: definitions central!

Communication over the Internet



ideal assumption

secure channel between Alice and Bob

crypto assumption

Adversary completely controls network! Can read/manipulate/drop all messages

reality

which assumption is realistic? which one is useful?

Crypto Protocols: Scenarios and Goals

protocol security goals

- privacy (secrecy)
- authentication
- message exchange
- online banking
- contract signing
- online shopping
- key exchange
- electronic elections
- crypto currencies
- secure remote access

…

clearly, there are overlaps

key questions

- what are the **precise** security requirements here?
- how do we formalize these?

Crypto Protocols: Special Issues

quote

Security protocols are three line programs that people still manage to get wrong.

Roger Needham

reasons

- protocols must be “successful” against adversary
- bugs hard to find: how do we “debug” security holes?

bug in protocol $\hat{=}$ possible attack!

consider

- security against all **possible** attacks needed
- security against all **known** attacks is not enough!

Protocol Execution: Levels of Abstraction

- logical errors

- buffer overruns, length checks …

- compiler errors, library bugs …

- operation system bugs

- hardware design error, short-term failures

“complete analysis:” obviously undecidable!

Distinction: In this Lecture

abstraction level we consider

- protocols on “Alice-and-Bob” level
- finding “logical attacks” on this level

motivation

- reality: attacks not on RSA, but on protocol/application
  - RSA security: discussed in crypto lecture!
- implementation errors “similar” to usual software development
  - buffer overflows
  - SQL injection
  - “normal” bugs

as always: exceptions!

analysis on logical level does not guarantee system security!

Distinction: Not in this lecture

Adversary: Two Aspects

dishonest components

network attacker completely controls the network

parties parties do not follow protocol

no clear separation

- similar consequences: messages may be fake
- possible reasons:
  1. network delivers wrong message
  2. Bob acts dishonestly

treat both aspects uniformly

assumption: there is a single adversary $\mathcal{A}$ controlling the network and all dishonest parties (who all work together)

security tradition

“maximally pessimistic assumptions”

Overview

Part I: Crypto Protocols

Foundations

Cryptography

Tools Against Powerful Adversaries

assumption

- attacker controls network
- dishonest parties

defence?

use cryptography! encryption, signatures, hash functions …

caveats

- encryption alone does not ensure security
- need shared keys, PKI, …

→ cryptographic infrastructure

- now: brief discussion of cryptographic primitives

Crypto in this Lecture: A Black Box

important

- no prior knowledge about cryptography assumed
- see crypto lecture by Prof. Wilke, or [KW11], or vast literature

cryptography aspects

- encryption (symmetric and asymmetric)
- signatures (symmetric (MAC) und asymmetric)
- hash functions
- random numbers

application

- “abstract” view: no concrete algorithms!
- security properties: “idealized”!

Cryptographic Primitives: Encryption

two cases

symmetric (AES, DES, …) single key k for encryption and decryption

- encrypt: $X \times K \to Y$
- decrypt: $Y \times K \to X$

$$\mathrm{dec}^{\mathrm{s}}_k\left(\mathrm{enc}^{\mathrm{s}}_k(x)\right) = x$$

asymmetric (RSA, ElGamal, …) key k_A for encryption, $\hat{k}_A$ for decryption

- encrypt: $X \times K \to Y$
- decrypt: $Y \times \hat{K} \to X$
- mapping: $\hat{}$: $K \to \hat{K}$

$$\mathrm{dec}^{\mathrm{a}}_{\hat{k}_A}\left(\mathrm{enc}^{\mathrm{a}}_{k_A}(x)\right) = x$$

security property in both cases

without decryption key: **no** information about plaintext x obtainable from ciphertext y.

Cryptographic Primitives: Signatures

two cases

symmetric (MAC) (CMAC, …) single key k_{AB} for "signing" and verification

- sign: $mac\colon X \times K \to T$
- verify: $test\colon X \times K \times T \to \{\texttt{ok}, \texttt{error}\}$

$$test(x, k, mac_k(x)) = \texttt{ok}$$

asymmetric key (RSA, ElGamal, …) $\hat{k}_a$ to sign, k_a to verify

- sign: $sig\colon X \times \hat{K} \to S$
- verify: $test\colon X \times K \times S \to \{\texttt{ok}, \texttt{error}\}$
- mapping: $\hat{}\colon K \to \hat{K}$

$$test(x, k, \mathrm{sig}_{\hat{k}}(x)) = \texttt{ok}$$

security property in both cases

without signing key: **impossible** to generate accepted signature t / s

Cryptographic Primitives: Hash Functions

perfect hash function

computation $hash : X \to T$

collision resistance if $x \neq y$, then $\mathsf{hash}(x) \neq \mathsf{hash}(y)$

information about message

- naive approach: $\mathsf{hash}(x)$ gives no information about x
- problem?
- attacker capabilities?: see later (equational theories)

Cryptographic Primitive: Random Generator

ideal properties

- random generators always return fresh values
- random bitstrings cannot be guessed

consequences

- session ids are perfectly random and unpredictable
- randomisation for encryption and other primitives is always perfect
- key generators are perfect

Exercise

Task (WhatsApp Authentication)

The instant messenges service WhatsApp for mobile phones uses the following authorization schemes:

1. To activate an account, the user needs to register a phone number. The system then sends a text message (SMS) over the mobile phone network to the user. The message contains a random number, which the user enters into the app. This activates the account.
2. To mirror the mobile app in a web browser, the user visits a special web page, which displays a QR code. The user then scans this code using the app, and can then access her account from the web interface.

Use informal notation and arguments to specify and discuss the security of the protocols underlying these authentication mechanisms. Think about whether encryption and/or signatures are used in the protocols, which (cryptographic) infrastructure is required to run the protocol, and which assumptions the protocol designers made.

Next Session: Review Questions

Next Session

review questions

- we will start the session with discussing review questions
- 5-15 minutes, depending on
  - time (I will roughly follow last year's schedule)
  - participation

your preparation

- review lecture notes up to today
- try to answer review questions marked "during semester"

your participation

- to have a nice discussion: activate cameras!
- come with follow-up questions or ideas for answers!
- present in class orally or via screen-sharing

before we go

any questions?

Thanks!

"See you" next time!

References

References i


Michael Backes, Birgit Pfitzmann, and Michael Waidner. “A General Composition Theorem for Secure Reactive Systems”. In: TCC. Ed. by Moni Naor. Vol. 2951. Lecture Notes in Computer Science. Springer, 2004, pp. 336–354. ISBN: 3-540-21000-8.


Ran Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: FOCS. 2001, pp. 136–145.


Ralf Küsters. “Simulation-Based Security with Inexhaustible Interactive Turing Machines”. In: CSFW. IEEE Computer Society, 2006, pp. 309–320. ISBN: 0-7695-2615-2.


Ralf Küsters and Thomas Wilke. Moderne Kryptographie - Eine Einführung. Vieweg + Teubner, 2011. ISBN: 978-3-519-00509-4.

References ii

Yehuda Lindell. “How to Simulate It - A Tutorial on the Simulation Proof Technique”. In: Tutorials on the Foundations of Cryptography. Ed. by Yehuda Lindell. Springer International Publishing, 2017, pp. 277–346. ISBN: 978-3-319-57047-1. DOI: 10.1007/978-3-319-57048-8_6. URL: https://doi.org/10.1007/978-3-319-57048-8%5C_6.

Thanh Binh Nguyen, Christoph Sprenger, and Cas Cremers. “Abstractions for security protocol verification”. In: Journal of Computer Security 26.4 (2018), pp. 459–508. DOI: 10.3233/JCS-15769. URL: https://doi.org/10.3233/JCS-15769.

Henning Schnoor. “Deciding Epistemic and Strategic Properties of Cryptographic Protocols”. In: ESORICS. Ed. by Sara Foresti, Moti Yung, and Fabio Martinelli. Vol. 7459. Lecture Notes in Computer Science. Springer, 2012, pp. 91–108. ISBN: 978-3-642-33166-4.

 Mathy Vanhoef and Frank Piessens. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. 2017.