Engineering Secure Software Systems

Winter 2020, Weeks $\approx 1-3$: Introduction and Formal Protocol Model

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

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

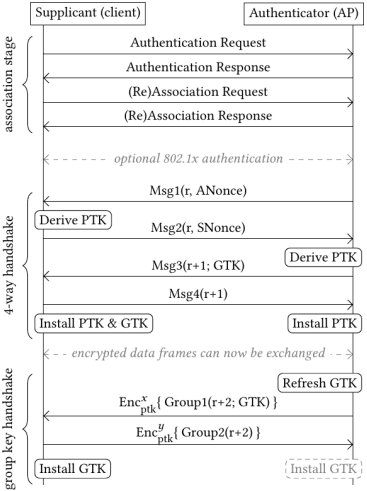
vulnerability

- 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

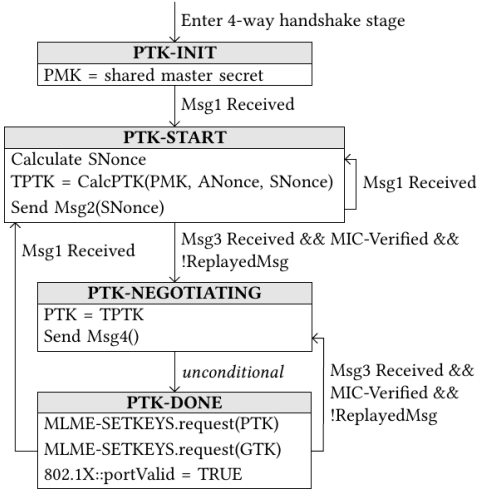
question

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

?

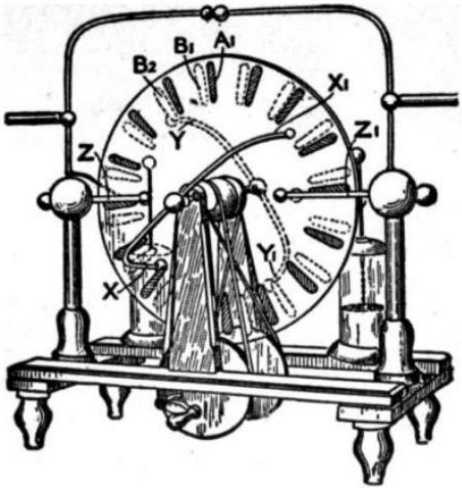
this lecture

- 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

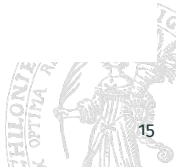
key topics

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

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

Overview

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

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

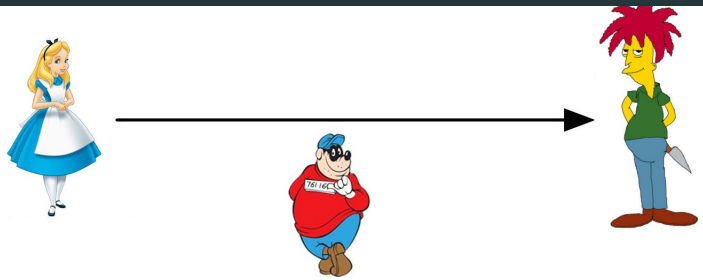
examples

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

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

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.

Assumptions too strong?

impossible goals with adversary-controlled network

- reply after $\leq t$ seconds
- reliable emergency call system
- delivery guarantee for messages
- …

in general

"liveness" properties cannot be guaranteed when network is completely unreliable

protocol design futile

If all others "maximally dishonest:" communication not reasonable

consequence: assumptions (depend on scenario), examples:

- at least Alice and Bob are honest
- existence of a trusted third party (TTP)
- Alice and Bob share secret key
- availability of PKI (public-key infrastructure)

Hopeless Situations?

cryptography can only help you so far …

political electronic elections in Germany

Der Zweite Senat hat entschieden, dass der Einsatz elektronischer Wahlgeräte voraussetzt, dass die ***wesentlichen Schritte der Wahlhandlung*** *und der Ergebnisermittlung vom Bürger* ***zuverlässig und ohne besondere Sachkenntnis überprüft werden können****. Dies ergibt sich aus dem Grundsatz der Öffentlichkeit der Wahl (Art. 38 in Verbindung mit Art. 20 Abs. 1 und Abs. 2 GG), der gebietet, dass alle wesentlichen Schritte der Wahl öffentlicher Überprüfbarkeit unterliegen, soweit nicht andere verfassungsrechtliche Belange eine Ausnahme rechtfertigen.*

Example: Authentication

goal

Bob expects “authenticated” message from Alice

problems

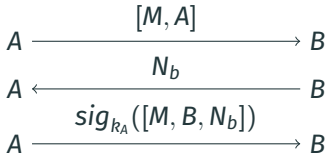
- attacker can always send message in Alice’s name!
- Bob needs way to check authenticity of message

cannot require that message arrives (liveness)

- require only: **if** Bob “accepts,” **then** message is from Alice.
- infrastructure: Alice “can do” something the attacker “can’t”
  - Alice has private key, Bob knows public key
  - Alice and Bob share private secret
  - Alice can authenticate herself using a certificate
  - …

A Secure Protocol: Simple Authentication

protocol



too complicated?

- why three messages?
- why is N_b needed?
- why must B be signed?

Bob's guarantees

- M comes from A (signed by A)
- M is intended for B (tuple containing M and B signed)
- M was confirmed **this session** (nonce N_b signed)

Exercise

Task (simple example protocol)

We consider the following simple authentication protocol:

- Alice sends a message M to Bob, together with her name A,
- Bob answers with a Nonce N_b,
- Alice answers with the term $\mathsf{sig}_{k_A}([M, B, N_B])$.

Please answer the following questions:

1. What are the security properties guaranteed by the protocol?
2. What is the purpose of the nonce N_B? What happens if we omit it?
3. What happens if the B is removed from Alice's last message?

consequences: “good practice” for protocol design

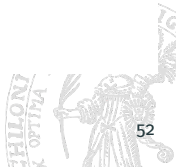
common-sense rules

- messages always contain sender and recepient
- mechanism to guarantee message “freshness”
  - “fresh” nonces with challenge-response
  - timestamps
- sign important components **together**
  - protocol above: **individual** signing of M and N_b not enough!

attention

following these rules neither sufficient not mandatory:

- there are secure protocols not following the rules
- there are insecure protocols following the rules



Overview

An Example: Authentication

goal

authentication: Bob wants to be sure that he is talking to Alice

infrastructure

PKI: Alice and Bob have public keys k_A and k_B

authentication protocol

$A \to B$	A	Hi, I'm Alice!
$B \to A$	$\mathsf{enc}^{\mathsf{a}}_{k_A}(N_B)$	Prove this!
$A \to B$	$\mathsf{enc}^{\mathsf{a}}_{k_B}(N_B)$	I know Alice's secret key $\hat{k}_A$!

secure protocol?

- can Bob be sure he is talking to Alice when he receives $\mathsf{enc}^{\mathsf{a}}_{k_B}(N_B)$?
- obvious "bug" in protocol?

The Needham-Schroeder Protocol

goal

authentication and key exchange

protocol

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

then $N_A \oplus N_B$ secure session key for A and B

really?

- protocol: 1978 [NS78]
- attack found: 1995 [Low96]

recall

Needham: three-line programs!

Attack on Needham-Schroeder

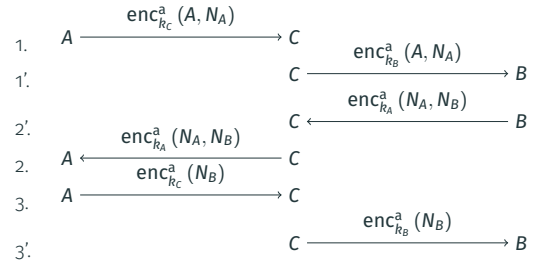
protocol

1. $A \to \cancel{B}\ C \quad \mathrm{enc}^{\mathrm{a}}_{\cancel{k_B}\ k_C}(A, N_A)$
2. $\cancel{B}\ C \to A \quad \mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, \cancel{N_B}\ N_C)$
3. $A \to \cancel{B}\ C \quad \mathrm{enc}^{\mathrm{a}}_{\cancel{k_B}\ k_C}(\cancel{N_B}\ N_C)$

situation

- Alice starts protocol as initiator with C (attacker)
- Bob starts protocol as responder with Alice
- adjust protocol for this situation

attack (Charlie controlled by $\mathcal{A}$)



consequence

- who is attacked?
- Bob "thinks" only Alice knows N_A and N_B
- C knows N_A and N_B

The Needham-Schroeder-Lowe Protocol

protocol

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

then $N_A \oplus N_B$ secure session key for A and B

intuition

- attack “mixes” messages from different protocol sessions
- consequence: B “talks to” C instead of A
- change: A realizes that message does not come from C

Attack on Needham-Schroeder-Lowe?

protocol

1. $A \to B \quad \mathrm{enc}^{\mathrm{a}}_{k_B}(A, N_A)$
2. $B \to A \quad \mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, N_b, B)$
3. $A \to B \quad \mathrm{enc}^{\mathrm{a}}_{k_B}(N_b)$

consequence

- Alice “talks to C,” receives message with “sender” B
- Alice aborts
- **good practice**: sender and receiver in messages

attack attempt

1. $A \xrightarrow{\mathrm{enc}^{\mathrm{a}}_{k_C}(A, N_A)} C$
1′. $C \xrightarrow{\mathrm{enc}^{\mathrm{a}}_{k_B}(A, N_A)} B$
2′. $C \xleftarrow{\mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, N_B, B)} B$
2. $A \xleftarrow{\mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, N_B, B)} C$
3. abort

Exercise

Task (Fixing Broken Authentication Protocols)

Consider the two authentication protocols presented in the exercise class:

a)

1. $A \to B \quad (A, \mathrm{enc}^{\mathrm{a}}_{k_B}(N_A))$
2. $B \to A \quad (B, \mathrm{enc}^{\mathrm{a}}_{k_A}(N_A))$

b)

1. $A \to B \quad (\mathrm{enc}^{\mathrm{a}}_{k_B}(N_A), \mathrm{enc}^{\mathrm{a}}_{k_B}(A))$
2. $B \to A \quad (\mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, N_B), \mathrm{enc}^{\mathrm{a}}_{k_A}(B))$

Both of these protocols can be attacked with a similar attack as the Needham-Schroeder protocol or the example protocol we covered in the first exercise class. Suggest changes to the protocols that address these problems, and argue why you think your revised versions of the protocols are secure. Be as specific as possible in what “secure” means in this case.

Overview

Example: Woo-Lam Authentication Protocol

prerequisites

k_{AS}, k_{BS}: symmetric keys shared between Alice (Bob) and Server

protocol

1. $A \to B \quad A$
2. $B \to A \quad N_B$
3. $A \to B \quad \mathrm{enc}^{\mathrm{s}}_{k_{AS}}(N_B)$
4. $B \to S \quad \mathrm{enc}^{\mathrm{s}}_{k_{BS}}\left([A, \mathrm{enc}^{\mathrm{s}}_{k_{AS}}(N_B)]\right)$
5. $S \to B \quad \mathrm{enc}^{\mathrm{s}}_{k_{BS}}(N_B)$

idea

- only Alice can encrypt N_B with k_{AS}
- server can check correctness

issues?

- server is “decryption oracle”
- Alice does not “know” that she “talks to Bob”

reference

Thomas Y. C. Woo and Simon S. Lam. “Authentication for Distributed Systems”. In: Computer 25.1 (Jan. 1992), pp. 39–52. ISSN: 0018-9162. DOI: 10.1109/2.108052. URL: http://dx.doi.org/10.1109/2.108052

Woo-Lam Protocol is insecure

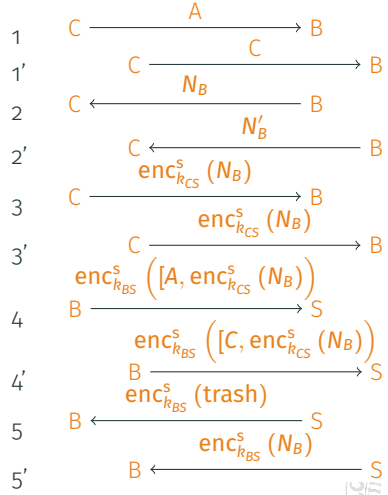
protocol

1. $A \to B \quad A$
2. $B \to A \quad N_B$
3. $A \to B \quad \mathsf{enc}^{\mathsf{s}}_{k_{AS}}(N_B)$
4. $B \to S \quad \mathsf{enc}^{\mathsf{s}}_{k_{BS}}\left([A, \mathsf{enc}^{\mathsf{s}}_{k_{AS}}(N_B)]\right)$
5. $S \to B \quad \mathsf{enc}^{\mathsf{s}}_{k_{BS}}(N_B)$

analysis

- **trash**: result of decrypting $\mathsf{enc}^{\mathsf{s}}_{k_{CS}}(N_B)$ with K_{AS}
- B believes A participated in protocol run
- assumptions?

attack: C controlled by $\mathcal{A}$, B honest



Overview

Overview

Finding Attacks

seen up to now: manual analysis

issues found by

- construction of message sequence: reachability of “bad state”
- argument that the attacker “knows” each message she sends: attacker gains sufficient knowledge to trigger required events

goal: automatic analysis

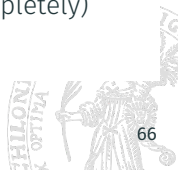
want algorithm that comes up with attack, or “proves” that there is no attack

Protocol Model: Components

goal
- formal security analysis
- manual + automatic
- **provable** security properties

needs
- formal model to reason about systems, security
- formalize protocols, their semantics + security

components of formal model
- messages
- message processing
- protocol specification: different roles (e.g., client, server, …)
- cryptography: honest participants, attacker (recall: attacker controls network completely)
- running sessions and scheduling (we have a distributed system)



Model Requirement: Express Needham Schroeder

minimal requirement for formal model

must be able to formalize Needham-Schroeder(-Lowe) protocol, attack, security.

attacker actions

1. C "talks to Bob in Alice's name" — steps $1', 2', 3'$
2. C makes Alice accept N_B instead of N_C — step 2
3. C exchanges data between sessions — all steps
4. C lets Alice and Bob wait — steps $1', 2', 2, 3$

consequences for model

1. untrusted message delivery
2. Alice's side of protocol must not mention N_C
3. attacker can send arbitrary terms, limited only by cryptography
4. attacker controls scheduling

Model Requirements: Generalizing from Needham-Schroeder Example

untrusted message delivery

messages delivered by network without meta-information

Alice's side of protocol must not mention N_C

expected terms cannot be hard-coded, model steps as receive/send-rules with variables instead

attacker can send arbitrary terms, limited only by cryptography

adversary controls network, uses "message construction" rules precisely defined using so-called Dolev-Yao closure

attacker controls scheduling

scheduling (execution order) explicitly done by adversary

Roadmap: Formal Model

features

1. untrusted message delivery
2. Alice's side of protocol must not mention N_C
3. attacker can send arbitrary terms, limited only by cryptography
4. attacker controls scheduling

addressed in formal definition

messages	formal terms
message construction, delivery, parsing	Dolev-Yao closure, receive/send actions, substitutions, matching
protocol specifications	protocol instance, protocol
scheduling	execution order
protocol security: no combination of adversary actions breaks protocol goal	protocol runs, (successful) attacks

Overview

Messages: Abstracting from Bitstrings

implemented protocol

- messages are bitstrings
- constructed by crypto algorithms
- attacker: arbitrary probabilistic polynomial-time algorithm

formal model

- messages are terms
- algorithms represented by function symbols
- attacker: nondeterministic choice of messages

why?

advantages of term model?

Definition: Messages as Terms

definition: terms

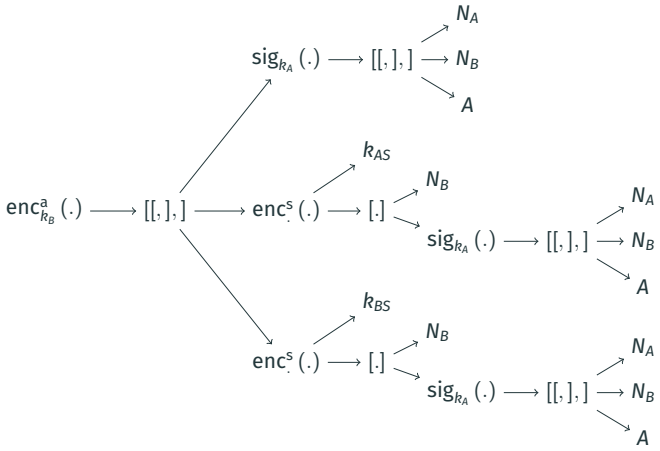
$\mathcal{T}$: smallest set with

- $\{\epsilon\} \cup \mathcal{C} \cup \mathcal{V} \cup \mathsf{IDs} \subseteq \mathcal{T}$, — empty message, constants, variables, names
- for all $i \in \mathbb{N}$, all $a \in \mathsf{IDs}$: $N_i, k_a, \hat{k}_a \in \mathcal{T}$, — random values, keys
- if $t_1, t_2 \in \mathcal{T}$, then $[t_1, t_2] \in \mathcal{T}$, — pairs/sequences
- if $t, t_k \in \mathcal{T}$, then $\mathsf{enc}^{\mathsf{s}}_{t_k}(t) \in \mathcal{T}$, — symm. encryption
- if $t \in \mathcal{T}$, $a \in \mathsf{IDs}$, then $\mathsf{enc}^{\mathsf{a}}_{k_a}(t) \in \mathcal{T}$, — asymm. encryption
- if $t, t_k \in \mathcal{T}$, then $\mathsf{MAC}_{t_k}(t) \in \mathcal{T}$, — symm. signature (MAC)
- if $t \in \mathcal{T}$, $a \in \mathsf{IDs}$, then $sig_{k_a}(t) \in \mathcal{T}$, — asymm. signature
- if $t \in \mathcal{T}$, then $\mathsf{hash}(t) \in \mathcal{T}$. — hash function

messages

term without variable: **ground term**, **message**.

Terms: Tree Representation



remarks

- converting to term representation is straight-forward
- optimization possibilities?

Definition: Subterms

definition: subterms

term $t \in \mathcal{T}$, then $\mathsf{Sub}(\mathsf{t})$ defined inductively:

- $\mathsf{Sub}(\mathsf{t}) = \{t\}$ if t atomic, i.e., $t \in \{\epsilon\} \cup \mathcal{C} \cup \mathsf{IDs} \cup \left\{ N_i, k_a, \hat{k}_a \mid i \in \mathbb{N}, a \in \mathsf{IDs} \right\}$
- $\mathsf{Sub}([\mathsf{t}_1, \mathsf{t}_2]) = \{[t_1, t_2], t_1, t_2\} \cup \mathsf{Sub}(\mathsf{t}_1) \cup \mathsf{Sub}(\mathsf{t}_2)$,
- $\mathsf{Sub}\left(\mathsf{enc}^{\mathsf{s}}_{\mathsf{t}_k}(\mathsf{t})\right) = \left\{\mathsf{enc}^{\mathsf{s}}_{t_k}(\mathsf{t}), t_k, t\right\} \cup \mathsf{Sub}(\mathsf{t}) \cup \mathsf{Sub}(\mathsf{t}_k)$,
- $\mathsf{Sub}\left(\mathsf{enc}^{\mathsf{a}}_{\mathsf{k}_a}(\mathsf{t})\right) = \left\{\mathsf{enc}^{\mathsf{s}}_{k_a}(\mathsf{t}), k_a, t\right\} \cup \mathsf{Sub}(\mathsf{t})$,
- $\mathsf{Sub}\left(\mathsf{sig}_{\mathsf{k}_a}(\mathsf{t})\right) = \left\{\mathsf{sig}_{k_a}(\mathsf{t}), k_a, t\right\} \cup \mathsf{Sub}(\mathsf{t})$,
- $\mathsf{Sub}(\mathsf{MAC}_{\mathsf{t}_k}(\mathsf{t})) = \{\mathsf{MAC}_{t_k}(\mathsf{t}), t_k, t\} \cup \mathsf{Sub}(\mathsf{t})$,
- $\mathsf{Sub}(\mathsf{hash}(\mathsf{t})) = \{\mathsf{hash}(\mathsf{t})\} \cup \mathsf{Sub}(\mathsf{t})$.

for $S \subseteq \mathcal{T}$: $\mathsf{Sub}(\mathsf{S}) = \cup_{t \in S} \mathsf{Sub}(\mathsf{t})$.

Overview

Model Requirement: Express Cryptographic Limitations

situation: $\mathcal{A}$ knows messages in set S

- own keys
- keys of dishonest parties
- "common knowledge" terms `init`, `request`, …
- messages sent by participants so far
- …

cryptographic operations to cover

• asymmetric encryption	always possible
• symmetric encryption	only with key
• decryption	only with key
• signature / MAC	only with key
• apply hash function	always possible
• …	

question

which messages can $\mathcal{A}$ send?

formally

define set $\mathsf{DY}(S)$ of messages that $\mathcal{A}$ can **derive** from S

Key Concept: Dolev-Yao Closure

reference

Danny Dolev and Andrew Chi-Chih Yao. “On the security of public key protocols”. In: IEEE Transactions on Information Theory 29.2 (1983), pp. 198–207

simple attacker modeling

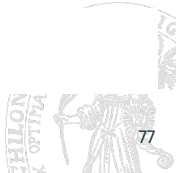
- standard model, many extensions
- consider primitives in isolation
- only derivations, no indistinguishability (see later)
- actual cryptography abstracted away

too simple?

- assume “perfect cryptography”
- practice: do RSA, AES, ElGamal satisfy this?
- abstraction step must be justified!

abstraction soundness

nontrivial topic, subtle issues — (possibly) later in the lecture!



Definition: Dolev-Yao Closure

$S \subseteq \mathcal{T}$, then $\mathrm{DY}(S)$ is the smallest set $D \subseteq \mathcal{T}$ with

- $S \cup \{\epsilon\} \cup \mathsf{IDs} \subseteq D$,
- $t_1, t_2 \in D$ iff $[t_1, t_2] \in D$,
- if $t \in D$ and $a \in \mathsf{IDs}$, then $\mathsf{enc}^{\mathsf{a}}_{k_a}(t) \in D$,
- if $t, t_k \in D$, then $\mathsf{enc}^{\mathsf{s}}_{t_k}(t), \mathsf{MAC}_{t_k}(t) \in D$,
- if $t \in D$ and $\hat{k}_a \in D$, then $\mathit{sig}_{k_a}(t) \in D$,
- if $\mathsf{enc}^{\mathsf{s}}_{t_k}(t) \in D$ and $t_k \in D$, then $t \in D$,
- if $\mathsf{enc}^{\mathsf{a}}_{k_a}(t) \in D$ and $\hat{k}_a \in D$, then $t \in D$,
- if $\mathsf{sig}_{k_a}(t) \in D$, then $t \in D$,
- if $\mathsf{MAC}_{k_i}(t) \in D$, then $t \in D$,
- if $t \in D$, then $\mathsf{hash}(t) \in D$.

intuition

- DY closure contains everything we cannot stop the adversary from knowing
- and nothing else!
- represents *optimistic* view of cryptography

note

model allows composed keys for symmetric cryptosystems

Dolev Yao Closure: Examples

initial adversary knowledge

$I = \left\{A, B, \hat{k}_I, k_{AI}, k_{BI}, 0, 1, \text{yes}, \text{no}\right\}$, knowledge grows with each message

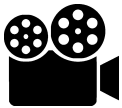
messages

$\mathcal{A}$ receives	goal	derivable?
$\text{enc}^{\text{a}}_{k_A}(\text{secret})$	secret	✗
$\text{enc}^{\text{a}}_{k_I}(\text{secret})$	secret	✓
$\text{enc}^{\text{s}}_{k_{AB}}(\text{yes})$	yes	✓
$\text{enc}^{\text{s}}_{k_{AB}}(N_A)$	N_A	✗
$\text{enc}^{\text{s}}_{k_{AI}}(k_{AB})$	N_A	✓
$\text{enc}^{\text{s}}_{k_A}([0,0,1,1,0,0,1,0,0,1,0,1,0,1,0,1])$	$[0,\ldots,1]$	✓
$\text{enc}^{\text{s}}_{[0,1,1,0,0,1,1,\ldots,0,1,1]}(N_B)$	N_B	✓

consequence

- arbitrarily long bit sequences always “known”
- do not model: adversary knows that **this message** contains “yes”
- $\rightsquigarrow$ generalization later

Overview



Computing the Dolev-Yao Closure

`https://cloud.rz.uni-kiel.de/index.php/s/No4yD7SPJaYz4wn`

video content

- characterization of $\mathsf{DY}(S)$ with *derivation rules*
- properties of “minimal derivations”
- a fixpoint algorithm for “computing” $\mathsf{DY}(S)$

study

- watch video—feedback welcome!
- video slides contained in slide set (gray background), additional material in lecture notes
- next week: discussion of content (in small groups), bring questions!

Goal: Compute Dolev-Yao Closure

Dolev-Yao

- proofs of insecurity (security): argue that adversary can (not) send message m
- need formal criterion of messages that adversary can send
- $\mathsf{DY}(S)$: set of messages the adversary can derive from S

long-term goal: automatic security analysis

need algorithm for $\mathsf{DY}(S)$

obstacle to computation

- $\mathsf{DY}(S)$ is infinite: ϵ, $[\epsilon, \epsilon]$, $[\epsilon, [\epsilon, \epsilon]]$, …
- algorithm cannot "write down" $\mathsf{DY}(S)$

way out

- we do not need to enumerate $\mathsf{DY}(S)$
- suffices to algorithmically answer question
  can adversary send m?

Result: Decision Procedure

decision problem

Problem:	DERIVE
Input:	set of terms S, term m
Question:	is $m \in \mathsf{DY}(S)$?

theorem

DERIVE can be decided in polynomial time.

reference

Michaël Rusinowitch and Mathieu Turuani. “Protocol insecurity with a finite number of sessions, composed keys is NP-complete”. In: Theoretical Computer Science 1-3.299 (2003), pp. 451–475

Technique: Proof Overview

steps

- characterization of Dolev-Yao Closure with **derivation rules**
- deciding whether $m \in \mathsf{DY}(S)$ is deciding whether there is a derivation of m from S
- issue: infinite search space of derivations
- solution:
  - if there is a derivation of m from S, then there is a shortest one
  - a shortest derivation contains no unnecessary steps
  - this restricts the search space

simplification

- to simplify case distinctions: only encryption, pairing, nonces, constants in this proof
- arguments suffice to also cover signatures, MACs and hash functions, …
- see exercise task for generalization

Tool: Derivation Rules

rules

for a message m, rule $L_d(m)/L_c(m)$ describes how m can be **decomposed**/**composed**

this potentially needs **prerequisites**:

- composing $\mathbf{sig}_{k_B}(m)$ needs m and $\hat{k}_B$
- decomposing $\mathbf{enc}^{\mathrm{s}}_{k_{AB}}(m)$ needs k_{AB}

a rule consists of a set R of **required** and and a set O of **obtained** terms, written $R \to O$. In the specific rules, we omit set brackets.

composition rules

$L_c([a,b])$	$a, b \to [a,b]$
$L_c(\mathrm{enc}^{\mathrm{a}}_{k_A}(m))$	$m \to \mathrm{enc}^{\mathrm{a}}_{k_A}(m)$
$L_c(\mathrm{enc}^{\mathrm{s}}_{t_k}(m))$	$m, t_k \to \mathrm{enc}^{\mathrm{s}}_{t_k}(m)$

decomposition rules

$L_d([a,b])$	$[a,b] \to a, b$
$L_d(\mathrm{enc}^{\mathrm{a}}_{k_A}(m))$	$\mathrm{enc}^{\mathrm{a}}_{k_A}(m), \hat{k}_A \to m$
$L_d(\mathrm{enc}^{\mathrm{s}}_{t_k}(m))$	$\mathrm{enc}^{\mathrm{s}}_{t_k}(m), t_k \to m$

Application of Rules: Derivations

definition

derivation: sequence $S_0 \to_{L_0} S_1 \to_{L_1} \dots S_{n-1} \to_{L_{n-1}} S_n$, such that, for all $i \in \{0, \dots, n-1\}$,

- L_i is a rule of the form $S \to S'$
- $S \subseteq S_i$
- $S_{i+1} = S_i \cup S'$

intuition

S_{i+1} obtained from S_i with L_i

Characterization: Dolev-Yao Closure Captured by Derivation Rules

lemma & definition

If $m \in \mathsf{DY}(S)$ where $\mathsf{IDs} \cup \{k_a \mid a \in \mathsf{IDs}\} \cup \{\epsilon\} \subseteq S$, then there is a derivation of m from S:

$$S = S_0 \to_{L_0} S_1 \to_{L_1} \dots S_{n-1} \to_{L_{n-1}} S_n$$

with $m \in S_n$. We call n the length of the derivation.

proof

see exercise

definition

For $m \in \mathsf{DY}(S)$, let $D_S(m)$ be a (fixed) shortest derivation of m from S. We write $L \in D_S(m)$, if L is a rule applied in $D_S(m)$.

Exercise

Task (DY closure and derivations)

In the lecture, the following lemma was stated (without proof):

If S is a set with $\mathsf{IDs} \cup \{k_a \mid a \in \mathsf{IDs}\} \cup \{\epsilon\} \subseteq S$ and $m \in \mathsf{DY}(S)$, then there is a derivation of m from S: $S = S_0 \to_{L_0} S_1 \to_{L_1} \ldots S_{n-1} \to_{L_{n-1}} S_n$ with $m \in S_n$.

1. Prove the above lemma.
2. State and prove an appropriate converse of the lemma.

Note: As in the lecture, you can assume that both S and m do not contain applications of hash functions, message authentication codes (MACs), or signatures.

Derivation Rules Property: One-Step Effects Only

composition

$L_c([a,b])$	$a, b \rightarrow [a,b]$
$L_c(\mathrm{enc}^{\mathrm{a}}_{k_A}(m))$	$m \rightarrow \mathrm{enc}^{\mathrm{a}}_{k_A}(m)$
$L_c(\mathrm{enc}^{\mathrm{s}}_{t_k}(m))$	$m, t_k \rightarrow \mathrm{enc}^{\mathrm{s}}_{t_k}(m)$

decomposition

$L_d([a,b])$	$[a,b] \rightarrow a, b$
$L_d(\mathrm{enc}^{\mathrm{a}}_{k_A}(m))$	$\mathrm{enc}^{\mathrm{a}}_{k_A}(m), \hat{k}_A \rightarrow m$
$L_d(\mathrm{enc}^{\mathrm{s}}_{t_k}(m))$	$\mathrm{enc}^{\mathrm{s}}_{t_k}(m), t_k \rightarrow m$

notation

$t \in \mathcal{T}$, then $\mathbf{Sub}^1(\mathrm{t})$ set of **direct subterms** of t

- $\mathbf{Sub}^1([\mathrm{t}_1, \mathrm{t}_2]) = \{t_1, t_2\}$
- $\mathbf{Sub}^1\left(\mathrm{enc}^{\mathrm{s}}_{\mathrm{k}}(\mathrm{t})\right) = \{k, t\}$
- $\mathbf{Sub}^1(\mathrm{hash}(\mathrm{t})) = \{t\}$
- …

direct successors in tree representation

observation

rules only work on $\mathbf{Sub}^1(.)$-level

- **composition rule** $L_c(m)$ has all terms from $\mathbf{Sub}^1(\mathrm{m})$ as prerequisites
- **decomposition rules** $L_d(m)$ obtains only terms from $\mathbf{Sub}^1(\mathrm{m})$

Reduce Search Space: Properties of Shortest Derivation

lemma

$D_S(m)$ shortest derivation of m from S, then:

1. If $L_d(t) \in D_S(m)$, then $t \in \mathsf{Sub}(\mathsf{S})$.
2. If $L_c(t) \in D_S(m)$, then $t \in \mathsf{Sub}(\mathsf{S} \cup \{\mathsf{m}\})$.

relevance

to derive m from S, we only need

1. decompositions of subterms from S
2. compositions of subterms of S or subterms of m

let's prove this!

written proof also contained in lecture notes

Exercise

Task (minimal derivation properties)

In the video lecture on the computation of the Dolev-Yao closure, we proved a lemma characterizing shortest derivations.

1. Can you generalize this result to handle signatures, MACs, and hash functions?
2. Which properties does the modeling of cryptographic primitives have to satisfy for an analog of this result to hold?
3. Can you come up with a modeling of cryptographic primitives where this property does not hold?

Algorithm for DERIVE

Input: set $S \neq \emptyset$ of messages, message m
  $S_{old} = \emptyset$
  while $S_{old} \neq S$ do
    $S_{old} = S$
    if ex. rule $S \rightarrow_L S \cup \{t\}$, $t \in \mathsf{Sub}(\mathsf{S} \cup \{\mathsf{m}\}) \setminus S$ then
      $S = S \cup \{t\}$
    end if
  end while
  if $m \in S$ then
    accept
  end if
  reject

algorithm uses previous results

- uses result on steps appearing in minimal derivations
- fixpoint algorihm: expands set S until fix point reached
- terminates in polynomial time since there are only polynomially many choices for t

covered in exercise

- algorithm correctness
- cannot “decompose first, compose later”

Exercise

Task (DY algorithm correctness)

Prove that the algorithm for computing the DY closure (in its decisional variant **DERIVE**) as stated in the lecture is correct and runs in polynomial time. As in the lecture, restrict yourself to terms without applications of hash functions, signatures, or message authentication codes (MACs).

Overview

Definition: Receive/Send Actions

definition (receive/send actions)

receive/send action: pair $(r, s) \in \mathcal{T} \times \mathcal{T}$, write $r \to s$.

example from Needham-Schroeder

Bob's rule: $\mathsf{enc}^{\mathsf{a}}_{k_B}(A, x) \to \mathsf{enc}^{\mathsf{a}}_{k_A}(x, N_b)$

- terms k_A, k_B, A: known
- rule assumes knowledge of $\hat{k}_B$
- term N_b: new nonce (generated by Bob)
- x: references Alice's nonce, repeated in Bob's response

variable handling

- variable x: stores (supposedly) nonce from Alice
- nonce (value of x) potentially used again later in protocol
- need to store terms from variables

recall

Bob's (Alice's) protocol description must not contain N_A (N_B, N_C, ...).

Definition: Substitutions

definition: substitutions

- substitution: function $\sigma\colon \mathcal{V} \to \mathcal{T}$ with $\sigma(x) \neq x$ for a finite number of x
- σ **ground substitution**, if $\sigma(x)$ message for all x with $\sigma(x) \neq x$.

intuition

- finite local memory of participants
- $\sigma(x) = x$: “unused” variable

extension to terms

for $t \in \mathcal{T}$, σ substitution, $\sigma(t)$ defined inductively:

- $\sigma(x)$ defined for $x \in \mathcal{V}$
- $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$

examples

$\rightsquigarrow$ board/notes

Matching: Applying Receive/Send Actions

situation in protocol run

- memory: substitution σ
- next action: $r \to s$
- incoming message: m

question: does message m "match" rule $r \to s$?

definition: matching

a term r matches with message m and substitution σ via substitution σ', if $\sigma'(r) = m$ and $\sigma'(x) = \sigma(x)$ for all x with $\sigma(x) \neq x$.

sequence

- receive: message m
- r/s action $r \to s$, r matches via σ'
- replace σ with σ'
- send term $\sigma'(s)$

Receiving and Parsing a Message

central step

how do protocol participants process an incoming message?

situation in protocol run

- stateful protocols: substitution σ already contains values for some variables
- linear protocols: participant's next r/s step is $r \to s$
- incoming message: m

Alice

- substitution: $\sigma(x) = N_B^1$, $\sigma(y) = y$
- next step: $(x, y, N_A^1) \to \mathsf{enc}^{\mathsf{a}}_{k_B}(y, N_A^2)$
- incoming message: $(N_B^1, (\mathtt{ok}, N_B^2), N_A^1)$

action:

- set $\sigma(y) = (\mathtt{ok}, N_B^2)$
- send $\mathsf{enc}^{\mathsf{a}}_{k_B}((\mathtt{ok}, N_B^2), N_A^2)$

Bob

- substitution: $\sigma(z) = N_A^1 \neq N_A^2$
- next step: $\mathsf{enc}^{\mathsf{a}}_{k_B}((\mathtt{ok}, N_B^2), z) \to \mathsf{enc}^{\mathsf{a}}_{k_A}(N_B^3)$
- incoming message: $\mathsf{enc}^{\mathsf{a}}_{k_B}((\mathtt{ok}, N_B^2), N_A^2)$

action:

incoming message cannot be parsed with receive/send rule, no reply message sent

Matching: Example

motivation

- matching: checks whether incoming term fits expectations
- expectations depend on
  - next rule in the protocol: receive/send rule from protocol
  - terms seen previously in protocol run: current substitution σ

example situation

- next receive/send rule:
  $$\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(x^1_A, N^1_A\right), \mathrm{sig}_{k_B}\left(x^2_A, y\right)\right) \rightarrow \mathrm{sig}_{k_A}\left(y, x^1_A, x^2_A, N^1_A, N^2_A\right)$$
- substitution:
  - $\sigma(x^1_A) = N^1_B$
  - $\sigma(x^2_A) = N^2_B$
  - $\sigma(y) = y$

reactions to incoming terms

matches? resulting substitution/reply?

- $\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(N^1_B\right), \mathrm{sig}_{k_B}\left(N^2_B, N_C\right)\right)$
- $\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(N^2_B, N^1_A\right), \mathrm{sig}_{k_B}\left(N^1_B, N_C\right)\right)$
- $\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(N^1_B, N^1_A\right), \mathrm{sig}_{k_B}\left(N^2_B, N_C\right)\right)$
- $\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(N^1_B, N^1_A\right), \mathrm{sig}_{k_B}\left(N^2_B, N^2_B\right)\right)$

Overview

Formal Protocol Definition

definition: protocol instance $\mathcal{I}$
sequence of actions

- $r_0 \to s_0$,
- $r_1 \to s_1$, with $\mathcal{V}(s_i) \subseteq \cup_{j \leq i} \mathcal{V}(r_i)$ for all i.
- $\ldots$,
- $r_{n-1} \to s_{n-1}$

example role

1. $\epsilon \to \mathsf{enc}^{\mathsf{a}}_{k_B}(A, N_A)$
2. $\mathsf{enc}^{\mathsf{a}}_{k_A}(N_A, x) \to \mathsf{enc}^{\mathsf{a}}_{k_B}(x)$

consequences for modeling

what kind of protocols can (can't) we express?

definition: protocol

protocol consists of

- instances $\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}$, and
- a finite set $\mathbf{I}$ of messages (the initial adversary knowledge).

example

formal representation of the Needham-Schroeder protocol

protocol

$$
\begin{array}{ll}
A \to B & \mathrm{enc}^{\mathrm{a}}_{k_B}(A, N_a) \\
B \to A & \mathrm{enc}^{\mathrm{a}}_{k_A}(N_a, N_b) \\
A \to B & \mathrm{enc}^{\mathrm{a}}_{k_B}(N_b)
\end{array}
$$

formalization

Alice

$$
\begin{array}{lcl}
\epsilon & \to & \mathrm{enc}^{\mathrm{a}}_{k_B}(A, N_A) \\
\mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, y) & \to & \mathrm{enc}^{\mathrm{a}}_{k_B}(y)
\end{array}
$$

Bob

$$
\mathrm{enc}^{\mathrm{a}}_{k_B}(A, x) \to \mathrm{enc}^{\mathrm{a}}_{k_A}(x, N_B)
$$

Overview

Modeling Protocols: Attack?

components: instances

- contain r/s actions
- example NSL: $\mathsf{enc}^{\mathsf{a}}_{k_B}([A, x]) \rightarrow \mathsf{enc}^{\mathsf{a}}_{k_A}([x, N_n])$
- hard-coded assumption: Bob replies to Alice

fixed by protocol

- actual “protocol” (r/s actions)
- participants and “roles”
  - Alice: initiator
  - Bob: responder

issue

- protocol as formalized cannot be attacked!
- need different situation to show attack …

example

formal representation of the Needham-Schroeder protocol with attacker

sessions

$A \to C(\mathcal{A})$ $\quad \mathrm{enc}^{\mathrm{a}}_{k_C}(A, N_A)$

$B \to A(\mathcal{A})$ $\quad \mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, N_B)$

$A \to C(\mathcal{A})$ $\quad \mathrm{enc}^{\mathrm{a}}_{k_C}(N_B)$

$\to$ Details on the black board!

formalization

Alice

$$\epsilon \to \mathrm{enc}^{\mathrm{a}}_{k_C}(A, N_A)$$
$$\mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, y) \to \mathrm{enc}^{\mathrm{a}}_{k_C}(y)$$

Bob

$$\mathrm{enc}^{\mathrm{a}}_{k_B}(A, x) \to \mathrm{enc}^{\mathrm{a}}_{k_A}(x, N_B)$$

Protocol Model

saw

formal protocol model, example for protocol run

crucial question

when is a protocol secure?

need: attacker model

- completely controls network
- can control participants (obtain their private keys)
- **also:** can start protocol sessions!

intuition

- Needham-Schroeder: attack when Alice “starts protocol with $\mathcal{A}$ (C)”
- to find attack: adversary must be able to “start protocol”
- also: attacker controls interleaving

our model: sessions (for now) part of the protocol

consequence?

Adversary: control over “running instances”

adversary controls

- who talks to whom?
  - one session $A \to C$, one session $C \to B$
- what happens when?
  1. 1. $A \to C$
  2. 1′. $C \to B$
  3. 2′. $B \to C$
  4. 2. $C \to A$
  5. 3. $A \to C$
  6. 3′. $C \to B$
- what does Charlie send?
  - C controlled by $\mathcal{A}$
  - no protocol instance started for Charlie
  - $\mathcal{A}$ uses network to “play Charlie”
  - Charlie ($\mathcal{A}$) can send “anything”

Executing Instances I

situation: instances given

1. Alice as initiator with Charlie ($\mathcal{A}$)
2. Bob as responder with Alice (played by $\mathcal{A}$)

adversary:

- $\mathcal{A}$ controls C (knows C's private key)
- $\mathcal{A}$ impersonates A in session with Bob

attack works only with this order

allow $\mathcal{A}$ to control order

execution steps

1. $A \to C$
2. $A \to B$
3. $B \to A$
4. $C \to A$
5. $A \to C$
6. $A \to B$

Execution Order

intuition

$P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$ protocol

- each instance $\mathcal{I}_j$ has $|\mathcal{I}_j|$ steps
- instance $\mathcal{I}_j$ activated "at most $|\mathcal{I}_j|$ times"
- order **inside** $\mathcal{I}_j$: fixed by protocol

definition: execution order

$P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$ protocol. An execution order for P is a sequence o over $\{0, \ldots, n-1\}$ such that each $j \in \{0, \ldots, n-1\}$ appears at most $|\mathcal{I}_j|$ times.

notation

- $o(t)$: t-th element in o
- $\#o(t)$: $|\{\ell \mid \ell < t \text{ and } o(\ell) = o(t)\}|$

position t

$\#o(t)$-th step of instance $\mathcal{I}_{o(t)}$

Possible Protocol Runs: Ingredients

given
- protocol $P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$
- execution order o of P

determines
- (honest) participants
- r/s actions

missing

actually delivered messages

possibilities
- distinguish between “honest” and “adversary” messages?
- for adversary:
  - messages intended to be sent by honest parties?
  - all messages $\mathcal{A}$ can “compute” from initial knowledge?
  - all messages $\mathcal{A}$ can “compute” from initial knowledge *and observed messages*?

Executing Instances II

situation
- attacker chooses instances, scheduling

assumption: $\mathcal{A}$ controls everyhing we don't.

next step: message delivery

step $A \to C$:
- belongs to instance "A with C ($\mathcal{A}$)"
- message created by protocol instance

step $A \to B$:
- belongs to instance "A ($\mathcal{A}$) with B"
- messages created by adversary

questions
- which messages can $\mathcal{A}$ send?
- restricted only by cryptography!
- what does this mean concretely?
- $\rightsquigarrow$ Dolev-Yao closure, up next

Overview

Protocol Runs: Formal Definition

definition: attack

$P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$ protocol, initial adversary knowledge I, i-th rule of $\mathcal{I}_j$ named $r_i^j \to s_i^j$. An attack on P consists of

- an execution order o for P,
- a substitution σ on the variables in P such that

$$\sigma(r^{o(k)}_{\#o(k)}) \in \mathsf{DY}\left(I \cup \left\{\sigma(s^{o(\ell)}_{\#o(\ell)}) \mid 0 \leq \ell < k\right\}\right).$$

simplification

- identify “protocol run” and “attack”
- protocol cannot be executed without $\mathcal{A}$ (network)

Successful Attack

next step

definition of successful attack

difficulty: different goals

- authentication
- key exchange
- secure channel
- electronic voting
- …

Successful Attack: Definition

variable "attack definition"

- success criterion for attack depends on protocol (goal)
- for automatic analysis: need protocol-independent definition
- integrate security definition into protocol: let designer specify security
- "program" instances so that $\mathcal{A}$ can get constant **FAIL** if successful (**BREAK**-rules)

definition: successful attack

$P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$ with initial adversary knowledge I. Attack (o, σ) is successful, if:

$$\mathsf{FAIL} \in \mathsf{DY}\left(I \cup \left\{\sigma(\mathsf{s}^{o(\ell)}_{\#o(\ell)}) \mid 0 \leq \ell < |o|\right\}\right).$$

literature: **secret** instead of **FAIL**

Application

example

application of definition to Needham-Schroeder protocol

steps

1. formalise protocol as r/s actions
2. formalise security specification: add **BREAK**-rules
3. find execution order for attack
4. find substitution for attack
5. check that attack is successful

→ Details on the black board!

Exercise

Task (Formal Representation of the Woo Lam Protocol)

Study the authentication protocol by Woo and Lam (see slide 22 of the lecture from October 29).

1. Specify the protocol as sequence of receive/send actions, once in the intended execution between Alice and Bob, and once in a form that allows to model the attack introduced in the lecture.
2. Specify the attack on the protocol formally.
3. How can we modify the protocol in order to prevent this attack?

Exercise

Task (Otway Rees Protocol)

Consider the following protocol (Otway-Rees-Protocol):

1. $A \to B \quad [M, A, B, \mathrm{enc}^{s}_{k_{AS}}([N_a, M, A, B])]$
2. $B \to S \quad [M, A, B, \mathrm{enc}^{s}_{k_{AS}}([N_a, M, A, B]), \mathrm{enc}^{s}_{k_{BS}}([N_b, M, A, B])]$
3. $S \to B \quad [M, \mathrm{enc}^{s}_{k_{AS}}([N_a, k_{AB}]), \mathrm{enc}^{s}_{k_{BS}}([N_b, k_{AB}])]$
4. $B \to A \quad [M, \mathrm{enc}^{s}_{k_{AS}}([N_a, k_{AB}])]$
5. $A \to B \quad \mathrm{enc}^{s}_{k_{AB}}(\mathrm{FAIL})$

1. Why are the subterms M, A, and B in the second message sent both encrypted and as plaintext?
2. Why is the nonce N_b encrypted in message 2?
3. Is the protocol secure? (You do not need to give a formal proof of security or insecurity.)

Successful Attack: Examples and Limits

examples

- key exchange
  - $\mathcal{A}$: [**BREAK**, k] to Alice
  - Alice replies with **FAIL**
- electronic voting
  - $\mathcal{A}$: [**BREAK**, $vote_B$] to Bob
  - Bob replies with **FAIL**

attention

- **BREAK**-messages distinguishable from “normal” messages: **FAIL** value used nowhere else
- see a “bug” in the examples?
- limits of this approach?

note

Voting example does not work like this! What else can we do?

Goal: Automatic Analysis

protocols formalizes

- protocol
- messages
- protocol execution
- successful attack

formal problem

Problem:	INSECURE
Input:	protocol P, initial adversary knowledge I
Question:	is there a successful attack on P?

algorithm for INSECURE?

is the problem decidable?

Exercise

Task (Security Modeling Issues: Are we Missing Something?)

In the lecture, we defined security of a protocol as, essentially, unreachability of a state in which the adversary learns the constant FAIL. However, this FAIL-constant obviously does not have a correspondance in a real implementation of a protocol. In particular, the rules releasing the FAIL-constant are removed from the protocol in a real implementation. As a consequence, a potential security proof of a protocol in our formal model treats a different protocol than the protocol running in a real implementation.

Are there cases where this difference results in an insecure protocol that can be proven secure in our formal model? If this is the case, how can we circumvent this issue?

Exercise

Task (Formal Protocol Model: Features and Omissions)

There are a couple of usually assumed properties of cryptographic systems that are not explicitly expressed in our protocol model. Which of the following properties are implicitly expressed in our model, and which are not? Are any of the “omissions” problematic?

- Nonces are indeed used only once, and are freshly generated for each session.
- Private keys are never sent over the network.
- There is a complete PKI available.
- Nonces are long enough so that the adversary cannot guess them correctly.
- The adversary knows the involved algorithms, including the protocol (Kerckhoffs’s principle).
- In the absence of an adversary, the network simply forwards the protocol participant’s messages as intended.

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.


Danny Dolev and Andrew Chi-Chih Yao. “On the security of public key protocols”. In: IEEE Transactions on Information Theory 29.2 (1983), pp. 198–207.


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.


Gavin Lowe. “Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR”. In: TACAS. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 1055. Lecture Notes in Computer Science. Springer, 1996, pp. 147–166. ISBN: 3-540-61042-1.


Roger M. Needham and Michael D. Schroeder. “Using Encryption for Authentication in Large Networks of Computers”. In: Communications of the ACM 21.12 (1978), pp. 993–999.

References iii


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.


Michaël Rusinowitch and Mathieu Turuani. “Protocol insecurity with a finite number of sessions, composed keys is NP-complete”. In: Theoretical Computer Science 1-3.299 (2003), pp. 451–475.


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.

Thomas Y. C. Woo and Simon S. Lam. “Authentication for Distributed Systems”. In: Computer 25.1 (Jan. 1992), pp. 39–52. ISSN: 0018-9162. DOI: 10.1109/2.108052. URL: http://dx.doi.org/10.1109/2.108052.