COL876: SPECIAL TOPICS IN FORMAL METHODS

# Formal verification of security protocols

Lecture 1, 24 July 2023

# COURSE OUTLINE & LOGISTICS

- Objective: To learn about formally modelling and verifying cryptographic protocols, and use specialized tools for the same
- Involves concepts from automata theory, algorithms, logic, and programming languages
- Ideally, you should have taken COL202 (Discrete Math) and COL352 (Automata & ToC), or some equivalent thereof
- If not, talk to me after class!

# COURSE OUTLINE & LOGISTICS

- Mostly following class notes, and other uploaded material
- Lecture notes and any ancillary material will be uploaded after class
- Office hours: by appointment only
- Tentative homework/exam and evaluation policy:
  - At most three assignments (~40%)
  - In-class quizzes + class participation (~10%)
  - A final project presentation (~50%)

# COURSE OUTLINE & LOGISTICS

- Option to do the project individually or in groups of two.
- Individual:
  - Read and present a paper to the class
  - Also submit a written report
  - A list of suitable papers will be provided, but can choose any that is relevant to the course

# COURSE OUTLINE & LOGISTICS

- Option to do the project individually or in groups of two.
- Group of two:
  - Use an automated tool to model a security protocol from an RFC/standard, and verify at least two properties
  - Present your model to the class and upload code to Github
  - A list of suitable protocols will be provided, but can choose any that is relevant to the course

# FORMAL VERIFICATION: WHAT?

- Today's systems are ubiquitous but increasingly complex
- Need absolute guarantees about behaviour; difficult to get just by software testing
- Enter *formal verification*!
  - Make an abstract mathematical model of system — ignore "irrelevant" details
  - Cast any desirable property as a mathematical formula
  - Verify that said formula holds of said model
  - Profit (? Hopefully!)

# SECURITY PROTOCOL: WHAT?

- Sequence of message exchanges to achieve some desirable goal
- Built upon various cryptographic schemes used for manipulating information with some guarantees
- Cryptographic schemes can be assumed to be "perfect"
  - Public key crypto and digital signatures have evolved enough to give us some basic assurances about secrecy, authenticity &c.
  - So we're going to ignore attacks on crypto: hash collisions, buffer overflow, side channel attacks &c.

# DO WE **REALLY** NEED VERIFICATION?

- The logic underlying the protocol could itself be flawed!
- Attacks due to incorrect protocol logic:
  - Impersonation of Trusted Platform Modules and/or owners
  - Breach of anonymity while using RFID e-passports
  - An e-voting protocol used by the government of Estonia…

# EXAMPLE 0

- On a public network, two people share a number $m$, which they want kept secret.

$$A \rightarrow B : m$$

$$B \rightarrow A : m$$

- Is this protocol secure? If A and B finish executing this protocol, can a malicious intruder I get to know $m$?

# EXAMPLE 0

- The network is *public*; obviously should not send $m$ in the clear
- Need to ensure secrecy via crypto mechanisms like encryption
- Even if "secret" is secured via crypto, if it is a constant or picked deterministically, replay attacks are possible!
  - Pick a randomly generated number

# ASSUMPTIONS

- A and B "honest principals": assumed to not intentionally compromise the protocol
- To honest principals, I is just any other entity on the network
  - They will communicate via the protocol with I, if required
- If a message of the wrong format is received, or none received at all? Up to the implementation!

# EXAMPLE 1

- On a public network, two people share a randomly generated value $m$, which they want kept secret.

$$A \rightarrow B : A, \mathsf{enc}(m, \mathsf{pk}(B))$$

$$B \rightarrow A : \mathsf{enc}(m, \mathsf{pk}(A))$$

- Is this protocol secure? If A and B finish executing this protocol, can a malicious intruder I get to know $m$?
- Perfect crypto; I can learn $m$ from enc($m$, $k$) only if I has the inverse of $k$
  - enc($m$, $k$) = enc($m'$, $k'$) $\Rightarrow$ $m = m'$ and $k = k'$: cannot "accidentally" learn secrets

# EX1: MAN IN THE MIDDLE

$A \longrightarrow B : A, \mathsf{enc}(m, \mathsf{pk}(B))$

$B \longrightarrow A : \mathsf{enc}(m, \mathsf{pk}(A))$

$A \longrightarrow \quad : A, \mathsf{enc}(m, \mathsf{pk}(B))$

$I \longrightarrow B : I, \mathsf{enc}(m, \mathsf{pk}(B))$

$B \longrightarrow I : \mathsf{enc}(m, \mathsf{pk}(I))$

$\longrightarrow A : \mathsf{enc}(m, \mathsf{pk}(A))$

# EXAMPLE 2

- So the previous version suffered a man-in-the-middle attack
- Easy fix: include the name of the sender inside the encryption.

$$A \rightarrow B : \mathsf{enc}((A, \mathsf{enc}(m, \mathsf{pk}(B))), \mathsf{pk}(B))$$
$$B \rightarrow A : \mathsf{enc}(m, \mathsf{pk}(A))$$

- Is *this* protocol secure? If A and B finish executing this protocol, can I get to know *m*?

# EX2: TYPE FLAW+MULTI-SESSION

$A \longrightarrow B : \mathrm{enc}((A, \mathrm{enc}(m, \mathrm{pk}(B))), \mathrm{pk}(B))$

$B \longrightarrow A : \mathrm{enc}(m, \mathrm{pk}(A))$

$\mathrm{A} \longrightarrow \quad : \{(\mathrm{A}, \{m\}_{\mathrm{B}})\}_{\mathrm{B}}$

$\mathrm{I} \longrightarrow \mathrm{B} : \{(\mathrm{I}, \{(\mathrm{A}, \{m\}_{\mathrm{B}})\}_{\mathrm{B}})\}_{\mathrm{B}}$

$\mathrm{B} \longrightarrow \mathrm{I} : \{(\mathrm{A}, \{m\}_{\mathrm{B}})\}_{\mathrm{I}}$

$\mathrm{I} \longrightarrow \mathrm{B} : \{(\mathrm{I}, \{m\}_{\mathrm{B}})\}_{\mathrm{B}}$

$\mathrm{B} \longrightarrow \mathrm{I} : \{m\}_{\mathrm{I}}$

$\longrightarrow \mathrm{A} : \{m\}_{\mathrm{A}}$

*“Security protocols are three-line programs that people still manage to get wrong”*

– Roger Needham

# PROTOCOL VERIFICATION: HOW?

- Abstract protocol into a formal model (automata, logic &c.)
  - Assume perfect cryptography
- Specify required security guarantees as mathematical properties over these abstract models
- Prove these properties hold, preferably by automated means

# SYMBOLIC MODEL: DOLEV & YAO, 1983

- Split each communication into a send and a receive
- I is essentially the network
  - Each send captured by I
  - Each receive assumed to come from I
- A send action need not have a corresponding receive action!

# DOLEV-YAO INTRUDER

- Intruder I cannot break encryption, but, on the public channel, can
  - see any message sent on the channel
  - block any message from reaching the intended recipient
  - re-route any message to any principal
  - masquerade as any principal and send messages in their name
  - initiate new communication according to the protocol
  - generate messages — according to some rules

# MODELLING MESSAGES

- Split each communication into a send and a (potential) receive
- But what about the messages sent and received?
- Messages are not bitstrings
  - Ignore extraneous details like headers, metadata &c.
- Modelled as symbolic terms from a *term algebra*.

$$t := m \mid \mathsf{f}(t_1, \ldots, t_k)$$

# MORE ABOUT MESSAGES

- When can an agent/intruder send a particular message term?
- When they can generate it, according to particular rules.
- Will only consider “well-formed” protocols
  - Honest principals can always generate whatever messages they need to send according to the protocol
  - Need to check correct generation only for the intruder

# PROOF RULES FOR MESSAGES

| | |
|---|---|
| $\dfrac{}{X \vdash m}\,\mathbf{ax}(m \in X)$ | $\dfrac{X \vdash k}{X \vdash \mathsf{pk}(k)}\,\mathbf{pk}$ |
| $\dfrac{X \vdash (t_1, t_2)}{X \vdash t_i}\,\mathbf{split}$ | $\dfrac{X \vdash t \quad X \vdash u}{X \vdash (t, u)}\,\mathbf{pair}$ |
| $\dfrac{X \vdash \mathsf{enc}(t, \mathsf{pk}(k)) \quad X \vdash k}{X \vdash t}\,\mathbf{adec}$ | $\dfrac{X \vdash t \quad X \vdash \mathsf{pk}(k)}{X \vdash \mathsf{enc}(t, \mathsf{pk}(k))}\,\mathbf{aenc}$ |

Proof system for a term algebra
with pairing and asymmetric encryption

# VERIFYING PROPERTIES

- Many properties involve looking for a proof using these rules
- *Passive intruder problem*: can the intruder violate some desired property just by observing traffic on the network?
- *Active intruder problem*: can the intruder violate some desired property by orchestrating DY-allowable behaviours and then observing the resulting network traffic?

# VERIFYING PROPERTIES

- Passive intruder problem merely checks abstract derivability
  - For simple systems, in PTIME
- Active intruder problem needs taking into account various sources of unboundedness (instantiations, number of parallel executions etc)
  - Undecidable in general
  - Often solved by restricting some source of unboundedness
- Tools to automate verification: ProVerif, Tamarin, DeepSec…