Security Procotols: Specification & Analysis

An introduction

Aim of this material

- show you how to model security protocols
- show you how to use a tool to analyze secrecy and authentication properties of such models
- show you limitations of modeling and analysis in relation to implemented protocols

Approach taken

- we will learn the "ASCII" notation for models of security protocols
- we use tool AVISPA for translating such ASCII models into executable models
- we see how AVISPA can analyze such models
- we feature industrial case study to demonstrate limitations of ASCII approach

Administrative matters

- you may access AVISPA as a web service http://www.avispa-project.org/web-interface/basic.php
- or you may freely download and install AVISPA from http://www.avispa-project.org/download.html
- rest of material assumes you can run AVISPA in one of these ways

Acknowledgement

- material presented here has taken, adapted or extended open-access material developed in the AVISPA project
- please send comments or feedback on material presented here to h.huth@imperial.ac.uk
- the tutorial, user manual, and other documentation of the AVISPA project may be found at http://www.avispa-project.org/

Security protocols

- are short communication programs between agents in a network
- these programs have variety of aims
- e.g. to exchange an encryption key as shared secret
- e.g. to ensure that one is communicating to intended agent and not to an attacker
- protocols often specified as ASCII models

ASCII models

- Specifying protocols in ASCII only requires a standard keyboard, no math symbols
- ASCII spec useful as a means of communicating some of the protocol intent
- But leaves many things implicit
- Not so useful for analyses that mean to guarantee secrecy of keys, authentication of an agent, etc

ASCII model: example

1. A --> B: {Na}_K
2. B --> A: {Nb}_K
3. A --> B: {Nb}_K1, where K1=Hash(Na, Nb)

- two agents A and B
- A sends Na, encrypted with key K, to B
- B sends Nb, encrypted with key K, to A
- A sends Nb, encrypted with key K1, to B

ASCII model: notation

1. A --> B: {Na}_K
2. B --> A: {Nb}_K
3. A --> B: {Nb}_K1, where K1=Hash(Na, Nb)

- {m}_K is message m encrypted with symmetric key K
- m, n is the concatenation of messages m and n in that order (e.g. Na, Nb above)
- Hash(m) is output of hash function Hash on input message m

ASCII model: critique

1. A --> B: {Na}_K
2. B --> A: {Nb}_K
3. A --> B: {Nb}_K1, where K1=Hash(Na, Nb)

- plus: very compact notation
- plus: intuitive
- minus: details unarticulated (e.g. hidden assumptions about agent knowledge)
- minus: unclear how to analyze raw spec
- minus: cannot express some real protocols

Exercise

1. A --> B: {Na}_K
2. B --> A: {Nb}_K
3. A --> B: {Nb}_K1, where K1=Hash(Na, Nb)

- study the above protocol
- then articulate what agents need to know, may belief, and have to be able to do in order for each protocol step to complete
- then discuss whether such knowledge, belief, and abilities can be assured in real implemenations

Another ASCII model

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- now study a protocol in greater detail
- this will allow us to appreciate that:
- protocols have intent
- goals of protocols help meet such intent
- knowledge and belief of protocol participants matters

Denning-Sacco shared key protocol

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- Intent: distribution of shared key Kab
- makes use of trusted server S
- Goal: A, B share symmetric key after run
- Goal: at most A, B, and S know Kab
- Goal: message received in step 3 sent (as encyr. sub-message) in same run in step 2

Intent versus goals

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- Intent: distribution of shared key Kab
- intent says informally what protocol wants to achieve
- Goal(s): e.g. only A,B,S know Kab
- goals are technical properties that, when true, validate that protocol meets intent

Exercise

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- Intent: distribution of shared key Kab
- Review the goals stated earlier on
- Then discuss to what extent the realization of these goals can help in assuring that a protocol run meets the above intent

First step in detail

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- A send first message, so A is initiator
- message sent to server S
- message content: A, B
- intended meaning: A wants to run protocol as initiator with B as responder
- names used as identity credentials, real implementation might use certificates

What's needed

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- Completion of first step requires:
- Agent A knows which trusted server S to use in this protocol run
- Agent A knows which agent B is responder
- NOTE: such knowledge may be mere belief and agent A may run protocol with wrong beliefs!

Second step in detail

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- S sends a message encrypted with key Kas (shared between A and S)
- encyrpted message contains four sub-messages: name of responder (B), key Kab to be shared by A and B, timestamp T, and
- "ticket" {Kab, A, T}_Kbs encrypted with key Kbs shared between B and server

What's needed

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- Completion of second step requires:
- Server S can generate fresh key Kab
- Server S can generate faithful timestamp T
- Server S shares keys with A (Kas), B (Kbs)
- Server can encrypt messages with such keys (i.e. has access to the encryption service, be it hardware or software)

Third step in detail

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- A sends to B the ticket {Kab, A, T}_Kbs
- NOTE: this ticket is ciphertext so it is hard to validate whether it was built as stated
- A does not possess the decryption key Kbs as this is shared between S and B only
- So A may need to do some integrity checks, see next slide

What's needed

1. A --> S: A, B
2. S --> A: {B, Kab, T, {Kab, A, T}_Kbs}_Kas
3. A --> B: {Kab, A, T}_Kbs

- A needs to make sure that the ticket is indeed the fourth sub-message of the message he received in step 2
- A knows Kas and so can decrypt message received in step 2
- A now needs to know exact data format of decrypted message in order to retrieve the correct ticket (i.e. the fourth sub-message)

What ASCII models taught us

- want a richer language for modeling protocol specifications
- such a language needs to support analyses of secrecy and authentication
- and it should allow us to model industrial scale protocols
- next lecture introduces such a language and it analysis tool