Extending Dolev-Yao with Assertions

R Ramanujam, Vaishnavi Sundararajan, S P Suresh

ICISS 2014, Hyderabad
December 18, 2014

Outline

Outline

The Dolev-Yao Model

- Useful for modelling agents' abilities in cryptographic protocols
- Message space viewed as term algebra $t := m \mid (t_1, t_2) \mid \{t\}_k$
- Intruder is the network – has access to any communicated message, but cannot break encryption

$$\frac{}{X \vdash t}\ ax\ \ (t \in X)$$

$$\frac{X \vdash (t_0, t_1)}{X \vdash t_i}\ split_i\ \ (i = 0, 1) \qquad \frac{X \vdash t_0 \quad X \vdash t_1}{X \vdash (t_0, t_1)}\ pair$$

$$\frac{X \vdash \{t\}_k \quad X \vdash inv(k)}{X \vdash t}\ dec \qquad \frac{X \vdash t \quad X \vdash k}{X \vdash \{t\}_k}\ enc$$

Figure: Term derivation rules, where X is a set of terms

More about Dolev-Yao

- Dolev-Yao treats terms as tokens
- Recepients 'own' terms, can pass them along in own name
- What if protocol uses certificates? (Should only be verified, but not owned)
- Common behaviour, especially in protocols involving authorization and delegation.
- Surely if it's that common, Dolev-Yao handles it?
- Yes, it does.

But...

Dolev-Yao expresses certification in the following ways:

But...

Dolev-Yao expresses certification in the following ways:

- Cryptographic devices – zero knowledge proofs, bit commitment etc.

But...

Dolev-Yao expresses certification in the following ways:

Not concise or readable

- Cryptographic devices – zero knowledge proofs , bit commitment etc.

But...

Dolev-Yao expresses certification in the following ways:

Not concise or readable

- Cryptographic devices – zero knowledge proofs , bit commitment etc.
- Ad-hoc methods – by tagging a term with an agent's name to indicate origin etc.

But...

Dolev-Yao expresses certification in the following ways:

Not concise or readable

- Cryptographic devices – zero knowledge proofs , bit commitment etc.
- Ad-hoc methods – by tagging a term with an agent's name to indicate origin etc.

Not general enough

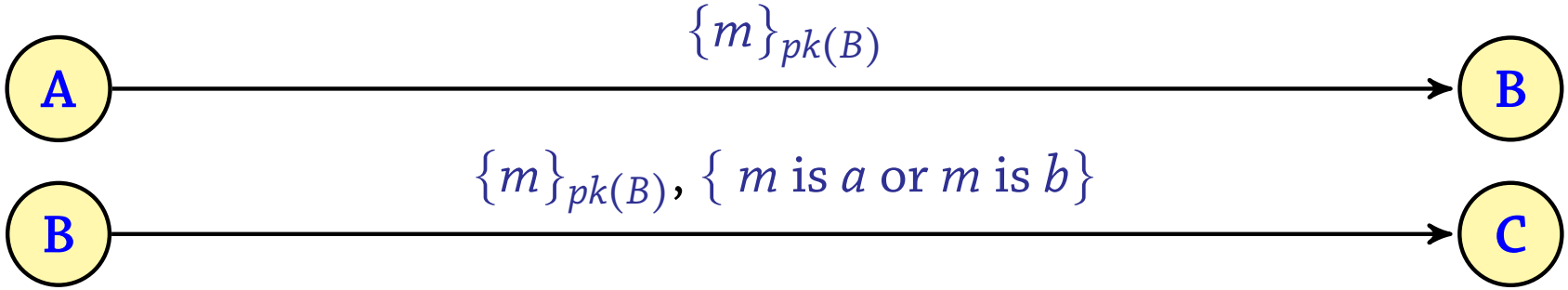
Outline

Example

We wish to model the following scenario.

Example 1

Agent A sends agent B a nonce m encrypted with B's public key. B then passes it on to a third agent C with some partial information about the value of m. (Suppose the actual value of m is a)



One of the most common ways to communicate such a certificate is by 1-out-of-2 re-encryption.

Modelling this in Dolev-Yao

1-out-of-2 Re-encryption: We have g and h known to everyone, where $h = g^s$ (s secret to the prover P). $enc(m) = (g^r, mh^r)$ is the term obtained on encrypting the term m with a random r. For a given pair (x, y), P must prove to V that it is of the form $\mathrm{enc}(m_i)$ where $m_i \in \{m_0, m_1\}$, without revealing i.

Modelling this in Dolev-Yao

1-out-of-2 Re-encryption: We have g and h known to everyone, where $h = g^s$ (s secret to the prover P). $enc(m) = (g^r, mh^r)$ is the term obtained on encrypting the term m with a random r. For a given pair (x, y), P must prove to V that it is of the form $\mathrm{enc}(m_i)$ where $m_i \in \{m_0, m_1\}$, without revealing i.

Basic idea:

- P sends to V the values $(x, y), \{m_0, m_1\}$.

Modelling this in Dolev-Yao

1-out-of-2 Re-encryption: We have g and h known to everyone, where $h = g^s$ (s secret to the prover P). $enc(m) = (g^r, mh^r)$ is the term obtained on encrypting the term m with a random r. For a given pair (x, y), P must prove to V that it is of the form $\mathrm{enc}(m_i)$ where $m_i \in \{m_0, m_1\}$, without revealing i.

Basic idea:

- P sends to V the values $(x, y), \{m_0, m_1\}$.
- V sends back $\left(x, \frac{y}{m_0}\right), \left(x, \frac{y}{m_1}\right)$.

Modelling this in Dolev-Yao

1-out-of-2 Re-encryption: We have g and h known to everyone, where $h = g^s$ (s secret to the prover P). $enc(m) = (g^r, mh^r)$ is the term obtained on encrypting the term m with a random r. For a given pair (x, y), P must prove to V that it is of the form $\text{enc}(m_i)$ where $m_i \in \{m_0, m_1\}$, without revealing i.

Basic idea:

- P sends to V the values $(x, y), \{m_0, m_1\}$.
- V sends back $\left(x, \frac{y}{m_0}\right), \left(x, \frac{y}{m_1}\right)$.
- P sends V a proof for 1-out-of-2 equality of discrete logarithm for $\left(x, \frac{y}{m_0}\right), \left(x, \frac{y}{m_1}\right)$.

Modelling this in Dolev-Yao +

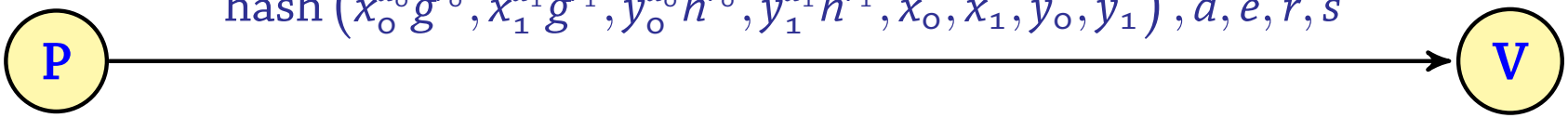
1-out-of-2 Equality of discrete logarithm (EDL): For fixed g and h known to everyone, and for given (x_0, y_0) and (x_1, y_1), the prover P must prove to V that there exist i and m such that $x_i = g^m$ and $y_i = h^m$.

Modelling this in Dolev-Yao +

1-out-of-2 Equality of discrete logarithm (EDL): For fixed g and h known to everyone, and for given (x_0, y_0) and (x_1, y_1), the prover P must prove to V that there exist i and m such that $x_i = g^m$ and $y_i = h^m$.

Choose d_0, d_1, r_0, r_1 randomly.
Set $c = \text{hash}\left(x_0^{d_0} g^{r_0}, x_1^{d_1} g^{r_1}, y_0^{d_0} h^{r_0}, y_1^{d_1} h^{r_1}\right)$.
Set $d_{1-i} = d$, $r_{1-i} = r$, $e = c - d$ and $s = md_i + r_i - me$.

What P does



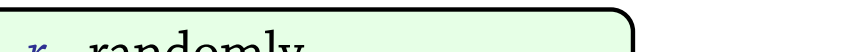
Check whether
$c \stackrel{?}{=} d + e \stackrel{?}{=} \text{hash}(x_{1-i}^d g^r, x_i^e g^s, y_{1-i}^d h^r, y_i^e h^s, x_0, x_1, y_0, y_1)$.

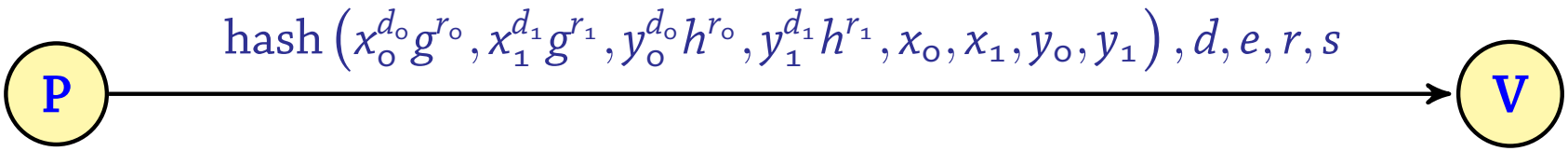
What V does

Modelling this in Dolev-Yao +

1-out-of-2 Equality of discrete logarithm (EDL): For fixed g and h known to everyone, and for given (x_0, y_0) and (x_1, y_1), the prover P must prove to V that there exist i and m such that $x_i = g^m$ and $y_i = h^m$.

Choose d_0, d_1, r_0, r_1 randomly.
Set $c = \text{hash}\left(x_0^{d_0} g^{r_0}, x_1^{d_1} g^{r_1}, y_0^{d_0} h^{r_0}, y_1^{d_1} h^{r_1}\right)$.
Set $d_{1-i} = d$, $r_{1-i} = r$, $e = c - d$ and $s = md_i + r_i - me$.





Check whether
$c \stackrel{?}{=} d + e \stackrel{?}{=} \text{hash}(x_{1-i}^d g^r, x_i^e g^s, y_{1-i}^d h^r, y_i^e h^s, x_0, x_1, y_0, y_1)$.

This clearly isn't very concise or readable.

Outline

What we want of assertions

An assertion should

- Be readable.
- Be non-ownable – agent B should not be able to send A's assertion in its own name.
- Be able to provide partial information about terms it references.
- Be communicated in a form which reveals the origin agent.

Assertion language

The set $\mathscr{A}$ of assertions is given by the following syntax

$$\alpha := m \prec t \mid t = t' \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2$$

where m is a nonce, and $m \prec t$ is to be read as m occurs in t.

Assertion language

The set $\mathscr{A}$ of assertions is given by the following syntax

$$\alpha := m \prec t \mid t = t' \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2$$

where m is a nonce, and $m \prec t$ is to be read as m occurs in t.

Disjunction allows us to model partial information certificates.

- In Example 1, the appropriate assertion is $a \prec \{m\}_{pk(B)} \vee b \prec \{m\}_{pk(B)}$. (Note that only one of the two disjuncts can be true at a time)

Communicated messages

In Example 1, the communication from B to C in the second step of the protocol looks as follows:

$$B \rightarrow C : \{m\}_{pk(B)}, \{a \prec \{m\}_{pk(B)} \vee b \prec \{m\}_{pk(B)}\}_{sd(B)}$$

The $sd(B)$ signifies that the assertion is signed by B. The communicated assertion thus carries information about the originating agent.

Example 1 with assertions

Our running example now reads as follows, when augmented with these assertions.

$$A \to B : \{m\}_{pk(B)}$$
$$B \to C : \{m\}_{pk(B)}, \{a \prec \{m\}_{pk(B)} \vee b \prec \{m\}_{pk(B)}\}_{sd(B)}$$

Example 1 with assertions

Our running example now reads as follows, when augmented with these assertions.

$$A \to B : \{m\}_{pk(B)}$$
$$B \to C : \{m\}_{pk(B)}, \{a \prec \{m\}_{pk(B)} \vee b \prec \{m\}_{pk(B)}\}_{sd(B)}$$

Much more succinct and readable than the Dolev-Yao version!

What about the intruder?

- The intruder I is still the network.
- But assertions, unlike terms, are signed. How does that affect I?
- I stores all signed assertions sent out, and may replay them later.

What about the intruder?

- The intruder I is still the network.
- But assertions, unlike terms, are signed. How does that affect I?
- I stores all signed assertions sent out, and may replay them later.
- Cannot modify assertions sent out earlier.
- Cannot replay an assertion by an agent in any other agent's name.

Why aren't there any proofs being sent in our version?

- Zero knowledge proofs put the burden of verification on recepients.
- Our paradigm: "perfect assertion assumption".
- Underlying system ensures only true assertions are sent out.
- Assertion's recepient no longer has to worry about checking its truth.
- Think of it as the underlying system being a verifying authority, and each agent sends a proof of its assertion to this authority. The authority checks the proof first, and allows the agent to send out the assertion only if the proof is correct.

Checks and derivations

When A sends a term t and an assertion α, the system checks that

- A can derive the term t from its set of terms X_A using Dolev-Yao rules.
- A can derive the assertion α from its set of assertions Φ_A using the system derivation rules (coming up on the next two slides).

Checks and derivations

When A sends a term t and an assertion α, the system checks that

- A can derive the term t from its set of terms X_A using Dolev-Yao rules.
- A can derive the assertion α from its set of assertions Φ_A using the system derivation rules (coming up on the next two slides).

When A receives assertion α (claiming to be) from B, the system checks that

- α is signed by B.
- B sent α out into the network earlier.

Derivation rules

$$\frac{X \vdash_{dy} m}{X, \Phi \vdash m \prec m}\ ax \qquad \frac{X \vdash_{dy} st(t) \cap \mathscr{B}}{X, \Phi \vdash t = t}\ eq$$

$$\frac{X \vdash_{dy} \{t\}_k \quad X \vdash_{dy} k \quad X, \Phi \vdash m \prec t}{X, \Phi \vdash m \prec \{t\}_k}\ enc \qquad \frac{X \vdash_{dy} inv(k) \quad X, \Phi \vdash m \prec \{t\}_k}{X, \Phi \vdash m \prec t}\ dec$$

$$\frac{X \vdash_{dy} (t_0, t_1) \quad X, \Phi \vdash m \prec t_i \quad X \vdash_{dy} st(t_{1-i}) \cap \mathscr{B}}{X, \Phi \vdash m \prec (t_0, t_1)}\ pair$$

$$\frac{X, \Phi \vdash m \prec (t_0, t_1) \quad X \vdash_{dy} st(t_i) \cap \mathscr{B} \quad m \notin st(t_i)}{X, \Phi \vdash m \prec t_{1-i}}\ split$$

Figure: The rules for atomic assertions

More derivation rules

$\dfrac{}{X, \Phi \cup \{\alpha\} \vdash \alpha}\ ax$	$\dfrac{X, \Phi \vdash m \prec \{b\}_k \quad X, \Phi \vdash n \prec \{b\}_k}{X, \Phi \vdash \alpha}\ \bot\ (m \neq n;\ b \in \mathscr{B})$
$\dfrac{X, \Phi \vdash \alpha_1 \quad X, \Phi \vdash \alpha_2}{X, \Phi \vdash \alpha_1 \wedge \alpha_2}\ \wedge i$	$\dfrac{X, \Phi \vdash \alpha_1 \wedge \alpha_2}{X, \Phi \vdash \alpha_i}\ \wedge e$
$\dfrac{X, \Phi \vdash \alpha_i}{X, \Phi \vdash \alpha_1 \vee \alpha_2}\ \vee i$	$\dfrac{X, \Phi \vdash \alpha_1 \vee \alpha_2 \quad X, \Phi \cup \{\alpha_1\} \vdash \beta \quad X, \Phi \cup \{\alpha_2\} \vdash \beta}{X, \Phi \vdash \beta}\ \vee e$

Figure: Rules for propositional reasoning

Outline

Derivability Problem and complexity

Derivability Problem: Given a set of terms X and a set of assertions Φ, and an assertion α, determine if $X, \Phi \vdash \alpha$ via the rules given earlier.

- This problem is co-NP-hard and in PSPACE.
- However, if we bound the number of disjunctions in α, the problem is solvable in PTIME.

Outline

What about forwarding?

Suppose B wants to forward an assertion α it received from A to agent C.

- Scenario is quite common in protocols employing delegation.
- We want to disallow B from just sending α in its own name.
- How to achieve this, then?

B sends C an assertion of the form A *says* α.

What about forwarding?

Suppose B wants to forward an assertion α it received from A to agent C.

- Scenario is quite common in protocols employing delegation.
- We want to disallow B from just sending α in its own name.
- How to achieve this, then?

B sends C an assertion of the form A *says* α.

Again, think of the underlying network as being a verifying authority. B basically tells the authority to approach A for a proof of α.

The set $\mathscr{A}$ of assertions is now given by the following syntax

$$\alpha := m \prec t \mid t = t' \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2 \mid A \text{ says } \alpha$$

Checks and derivations for *says*

On receiving α from A, B adds $A \ says \ \alpha$ to its assertion set Φ_B. Other checks and updates remain the same.

$$\frac{X, \Phi \vdash A \ says \ (m \prec \{b\}_k) \quad X, \Phi \vdash A \ says \ (n \prec \{b\}_k)}{X, \Phi \vdash A \ says \ \alpha} \perp \ (m \neq n; \ b \in \mathscr{B})$$

$$\frac{X, \Phi \vdash A \ says \ \alpha_1 \quad X, \Phi \vdash A \ says \ \alpha_2}{X, \Phi \vdash A \ says \ (\alpha_1 \wedge \alpha_2)} \wedge i \qquad \frac{X, \Phi \vdash A \ says \ (\alpha_1 \wedge \alpha_2)}{X, \Phi \vdash A \ says \ \alpha_i} \wedge e \qquad \frac{X, \Phi \vdash A \ says \ \alpha_i}{X, \Phi \vdash A \ says \ (\alpha_1 \vee \alpha_2)} \vee i$$

$$\frac{X, \Phi \vdash A \ says \ (\alpha_1 \vee \alpha_2) \quad X, \Phi \cup \{A \ says \ \alpha_1\} \vdash A \ says \ \beta \quad X, \Phi \cup \{A \ says \ \alpha_2\} \vdash A \ says \ \beta}{X, \Phi \vdash A \ says \ \beta} \vee e$$

Figure: Rules for *says*

Outline

Conclusion and future work

- Described a framework to add assertions to the Dolev-Yao model.
- Makes for concise and more readable certification in protocols.
- Also have key complexity results about this model.
- Future work: better assertion structure, tighter complexity bounds etc.

Thank you!