Applying Model Checking to Internet Security

Andrew Gusty, Undergraduate, Dept of Computer Science

Team ID: 13

University of Colorado - Theory of Computation

Abstract

Model checking is an important practice in engineering and software development to rigorously verify that systems operate as intended. In this project, I give an overview of model checking and its relevance to Linear Temporal Logic (LTL) and Linear Temporal Logic of Knowledge (LTLK). To illustrate model checking via LTL, I present an analysis of the Needham-Schroeder public-key protocol to reproduce the error originally found by Gavin Lowe.

Basics of Model Checking

Definition 1.1. Invariant Properties An LT property P_{inv} over AP is called an *invariant* if there is a propositional logic formula Φ over AP such that

$$P_{inv} = \{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall j \geq 0 \rightarrow A_j \models \Phi\}$$

That is, that Φ *holds for all time*.

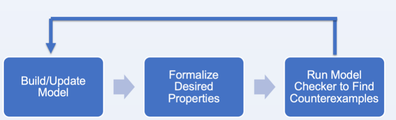
Definition 1.2. Safety Properties An LT property P_{safe} over AP is called a *safety* property if for all words $\sigma \in (2^{AP})^\omega \backslash P_{safe}$ there exists an infinite prefix $\hat{\sigma}$ of σ s.t.

$$P_{safe} \cap \{\sigma' \in (2^{AP})^\omega | \hat{\sigma} \text{ is a finite prefix of } \sigma'\} = \emptyset.$$

A safety property asserts that *nothing bad every happens*.

Definition 1.3. Liveness Properties An LT property P_{safe} over AP is called a *liveness* property whenever $pref(P_{live}) = (2^{AP})^*$

A liveness property asserts that each finite word can be extended to an infinite word that satisfies P, i.e. that *eventually something good will happen*.



Linear Temporal Logics

Definition 2.1 (Syntax of LTL)**.** Let AP be a set of atomic propositions.

- Every $p \in AP$ is an LTL formula (over AP).
- If f and g are LTL formulae, then so are $\neg f$, $f \vee g$, $f \wedge g$, $\bigcirc f$, $\diamond f$, $\square f$, $f \,\mathcal{U}\, g$, and $f \,\mathcal{R}\, g$

Operators associate to the right and unary operators have precedence over binary ones.

For some infinite word accepted by an LTL, σ at a given time-step, $i \in \mathbb{N}$, and some operator $\mathcal{L}$, we use the syntax $\sigma_i \models f$ to say that "f holds at position i of σ".

2.1 Linear Temporal Logic of Knowledge

For an LTL of knowledge, we use the same definition, but with the addition of *Agents*, $Ag = \{1, ..., n\}$. We use the notation $K_k\Phi$ to denote that "Agent k knows the LTL formula Φ".

Definition 2.2. (Syntax of Kripke Structure) For a non-empty set of atomic propositions, AP, A Kripke Structure is 4-tuple $M = < S, s_0, R, L >$

- S is a finite set of states
- $s_0 \in S$ is the initial state
- $R \subseteq S \times S$ is the transition relation, for which $\forall s \in S : \exists s'$, then $(s, s') \in R$
- $L : S \rightarrow 2^{AP}$ is a labeling function, which labels each state with the atomic propositions which hold in that state

The paths generated by a Kripke structure, starting at state $s \in S$ are infinite, and so can be used to see if they satisfy an LTL (or equivalently if they are accepted by an ω-automata). We use π^i, where i is the starting index of the path $\pi^i = s_i, s_{i+1}, ...$

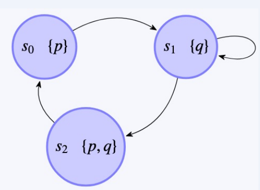


Figure 1: Kripke structure example

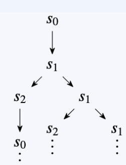


Figure 2: Tree diagram of possible paths starting from s_0

The Needham-Schroeder Vulnerability

The Needham-Schroder protocol is widely used on the internet to provide mutual authentication for two parties communicating on an insecure network. Agent A initiates a connection to agent B, who then responds to A. Messages are in the form $\{X, Y\}_{pub\_key(Z)}$, where X and Y are messages encrypted with Z's public key. A shorthand of the protocol can be described as,

$$\begin{aligned} &(1) \quad A \rightarrow B : \{N_A, A\}_{K_B} \\ &(2) \quad B \rightarrow A : \{N_A, N_B\}_{K_A} \\ &(3) \quad A \rightarrow B : \{N_B\}_{K_B} \end{aligned}$$

Safety Conditions

If A successfully completes a run with B, then I should not know A's nonce or B's nonce.

S1: $\square$(Running(A, B) $\rightarrow \neg K_I$ val_nonce(N_A, V))

S2: $\square$(Running(B, A) $\rightarrow \neg K_I$ val_nonce(N_A, V))

For all moments except the first moment, if M_1 is a message which contains N_1 and an agent knows the contents of N_1, then they already knew the content or received an encrypted M_1 that they could decode.

S3: $\forall X, M_1, N_1, \exists V_1 K_Xval_nonce(N_1, V_1) \iff \ominus$((Msg($M_1$) $\wedge$ contains(M_1, N_1))$\vee K_X$val_nonce(N_1, V_1)$\vee$($\exists Y, V.$ recv(X, M_1, pub_key(Y)) $\wedge K_X$ val_priv_key(Y, V)))

If an agent send a message M_1 encrypted with Key, then either knows the contents of M_1 or is simply forwarding it.

S4: $\forall X$, Key, M_1, N_1.(send(X, M_1.Key) $\wedge$contains(M_1, N_1)) $\rightarrow \exists V_1.K_X$val_nonce($N_1, V_1$) $\vee \blacklozenge$rcv(X, M_1,Key)

If an agent receives a message, then there was some agent that previously sent that message.

S5: $\forall X$, Key, M_1.rcv(X, M_1, Key) $\rightarrow \exists Y. \blacklozenge$send($Y, M_1$,Key)

Liveness Condition

Wherever both A and B have successfully completed a run of the protocol, then A should believe her partner to be B iff B believes to talk to A.

L1: $\square$(Running(A, B) $\rightarrow$ (Commit(A, B) $\iff$ Commit(B, A)))

Man-in-the-Middle Run

$$\begin{aligned} &(1) \quad A \rightarrow I : \{N_A, A\}_{K_I} \qquad \textbf{I learns } N_A \\ &(I1) \quad I(A) \rightarrow B : \{N_A, A\}_{K_B} \\ &(I2) \quad B \rightarrow I(A) : \{N_A, N_B\}_{K_A} \\ &(2) \quad I \rightarrow A : \{N_A, N_B\}_{K_B} \\ &(3) \quad A \rightarrow I : \{N_B\}_{K_I} \qquad \textbf{I learns } N_B \\ &(I3) \quad I(A) \rightarrow B : \{N_B\}_{K_B} \end{aligned}$$

Predicates of Needham-Schroeder Protocol:

- send(A, Msg, Key) is satisfied if agent A sends message Msg encrypted by Key.
- rcv(A, Msg, Key) is satisfied if agent A receives message Msg encrypted by Key.
- Msg(M_1) is satisfied if M_1 is a message.
- $\text{val}_{\text{pub\_key}}(X, V)$ is satisfied if the value of the public key of X is V.
- $\text{val}_{\text{priv\_key}}(X, V)$ is satisfied if the value of the private key of X is V.
- $\text{val}_{\text{nonce}}(N_A, V)$ is satisfied if the value of nonce N_A is V.
- contains(M_1, M_2) is satisfied if the message M_2 is contained within M_1.

Conclusions

This project presents a formal approach to model-checking using LTL and applies this approach to internet security protocols using LTL of Knowledge. Future work entails applying these methods to more complicated protocols using model-checking software such as SPIN or FDR. Additionally, my full project paper will include the formal theory of Büchi automata and its equivalence to LTL, as well as epistmic automata which recognize LTL of Knowledge.

References

[1] Baier, C., & Katoen, J.-P. (2008). Principles of model checking / Christel Baier, Joost-Pieter Katoen. MIT Press.

[2] Büchi, Richard. (1990). On a Decision Method in Restricted Second Order Arithmetic. The Collected Works of J. Richard BÃŒchi ISBN: 9781461389309.

[3] Lowe, G. (1996). Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR (Vol. 1055). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-61042-1_43

[4] Long Shigong, & Wang Lijun. (2010). Analysis of cryptographic protocols using LTL of knowledge. 2010 International Conference on Networking and Digital Society, Networking and Digital Society (ICNDS), 2010 2nd International Conference On, 1, 463-466. https://doi.org/10.1109/ICNDS.2010.5479238

[5] Vardi, M. Y. (2007). 17 Automata-theoretic techniques for temporal reasoning. Studies in Logic and Practical Reasoning, 3, 971-989. https://doi.org/10.1016/S1570-2464(07)80020-6