Closing the Idealization Gap with Theory Generation (Extended Abstract)

Darrell Kindred Jeannette M. Wing
Computer Science Department
Carnegie Mellon University
$\{$*dkindred,wing*$\}$*@cs.cmu.edu*

August 1997

1 Overview

Cryptographic protocol design demands careful verification during all phases of development. Belief logics, in the tradition of the Burrows, Abadi, and Needham (BAN) logic of authentication [BAN90], provide a simple, intuitive model, and allow natural expressions of a protocol and its goals. Since manual deduction is error-prone, protocol designers need automated tools to make effective use of these logics. Such tools often require excessive human intervention or supply inadequate feedback during the verification process.

We take a new approach, "theory generation," which allows highly automated reasoning with these logics, and which supports new forms of protocol analysis. In this approach, given a logic, L, we generate a finite representation, T^*, of the full theory, corresponding to a protocol, P. Given this representation, determining whether the protocol satisfies some property, ϕ, requires only a simple membership test, $\phi \in T^*$? (Figure 1). Furthermore, since the theory is represented by a finite set of formulas, we can analyze differences between protocols by comparing the generated theories, and we can easily answer questions such as, "What beliefs does this principal hold after receiving message 2?" In earlier work described in our USENIX paper, we applied theory generation to three different belief logics (BAN, AUTLOG [KW94], and Kailar's accountability logic [Kai96]), and seven protocols for authentication and electronic commerce [KW96].

BAN-style belief logics enable the designer to think about a protocol at a convenient level of abstraction; however, the gap between the "idealized" protocol

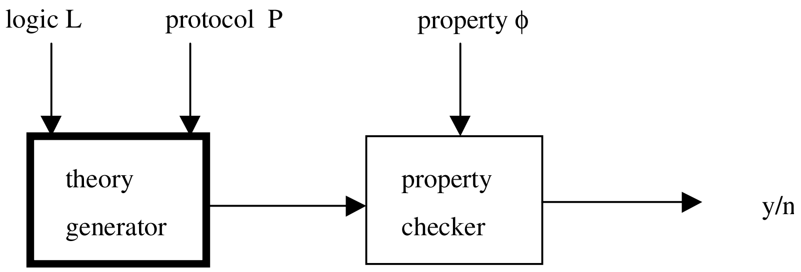

Figure 1: Basic Approach

and a concrete protocol implementation is substantial. In this paper, we show how, in the context of theory generation, to bridge this gap.

Many of the well-known deficiencies of these belief logics stem from this troublesome "idealization" step, in which the person verifying a protocol must devise an abstract interpretation for each of the concrete messages exchanged in the protocol. It is difficult to produce a good idealization. For example, the original BAN analysis of the Needham-Schroeder public-key authentication protocol failed to expose a flaw recently discovered by Lowe, in part because it used an idealization that masked the problem [Low95]. Although general guidelines for building idealizations exist, the idealization process itself is informal, and thus there is no way to reason formally about the validity of an idealization. An improper idealization may assign meanings to a message that the sender did not intend, making any further reasoning invalid. Even a "good" idealization may obscure important aspects of the protocol.

In this paper, we present a formalization of the idealization step (Section 3). We give a set of simple properties that will be required of legitimate idealizations, and then describe some more interesting properties that should hold for a given idealization and concrete protocol together.

Given a concrete protocol and formalized idealization, we can reason about four classes of properties, all within the context of theory generation:

- **belief goals**: These are the properties traditionally verified in BAN-style analysis, regarding beliefs held at the end of a protocol run [KW96].
- **honesty**: These properties assert that principals sign only messages that correspond to their beliefs.
- **responsibility**: These properties prevent principals from acting in manners that indirectly contradict their beliefs.

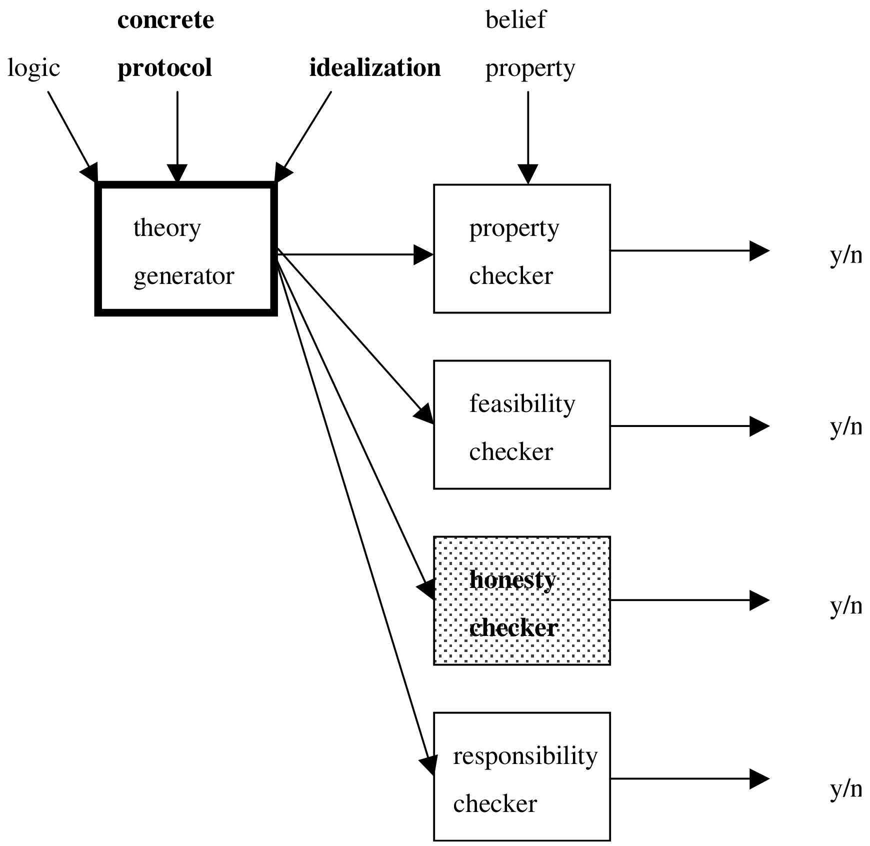

Figure 2: Fuller Approach

- **feasibility**: These properties ensure that principals have enough knowledge to carry out their assigned roles in the protocol.

In this paper we focus on *honesty* properties (Section 4).

Figure 2 gives a fuller schematic of our entire approach. Using the same theory generator, we can perform more than just the belief-property checks as suggested in Figure 1. Note that we need to modify the description of the protocol given as input: rather than give an abstract protocol, a user gives both a concrete protocol and a set of rules for assigning idealized meanings to concrete messages.

2 The Theory Generation Method[1]

The technique for checking whether some desired property, ϕ, holds for a given protocol, P, consists of these basic steps:

- Given a logic, L (consisting of a finite set of axioms and controlled-growth rules), our tool automatically generates a checker, C, specialized to that logic.
- Given the protocol, P, for which we want to check property ϕ, encode the initial assumptions and messages of P as formulas of L; call this set T^0.
- Using C, exhaustively enumerate the theory, T^*, that is, the set of facts (formulas) derivable from the formulas in T^0.
- Determine whether ϕ is in T^* by a simple membership test.

2.1 Class of Logics

We impose several restrictions on the logic, L, to ensure that this technique can be applied successfully.

- V is a formula for any variable V, and
- if $F_1, \ldots, F_n$ are formulas and S is a function symbol, then $S(F_1, \ldots, F_n)$ is a formula (for any $n \geq 0$).

A constant can be represented as a function symbol with no arguments (we will omit the parentheses in this case). In particular, no conjunction, disjunction, negation, or quantifiers are permitted. Here are some legal formulas:

$$\begin{array}{l} believes(A, shared\_key(K_{AB}, A, B)) \\ sees(B, encrypt(K_{BS}, shared\_key(K_{AB}, A, B))) \\ controls(S, shared\_key(K_{AB}, A, B)) \end{array}$$

We must be able to separate the rules of inference of the logic L into three classes:

[1]Most of the this section is straight from our USENIX paper. We include it to make this paper more self-contained.

- *S-rules*: An S-rule (shrinking rule) consists of a set of premises, $\{P_1, \ldots, P_n\}$, and a conclusion C, such that if $P_1, \ldots, P_n$ unify simultaneously with a set of derivable formulas, then C (appropriately instantiated) is a derivable formula. The conclusion must be the same size as or smaller than one of the premises, by some well-founded measure. Each variable occurring in the conclusion must also occur in one or more premises.

- *G-rules*: A G-rule (growing rule) has the same form as an S-rule, but the conclusion must be strictly larger than each of the premises, by the same measure, and each variable occurring in the premises must also occur in the conclusion.

- *Rewrites*: A rewrite is a pair of formulas, (f_1, f_2), such that any occurrence of f_1 in a derivable formula may be replaced by f_2, yielding another derivable formula. The formula f_2 must be the same size as f_1 and contain the same variables.

Finally, we require that each S-rule must still meet the S-rule criteria when all its premises that could match G-rule conclusions are removed. Whereas the other restrictions are local properties of individual rules and thus can be checked for each rule in isolation, this global restriction involves the whole set of rules.

2.2 Summary of Applications of Our Method

In our USENIX paper, we describe the theory-generation algorithm and present informal arguments for its correctness and termination. We explain how we applied it to three different logics and checked authentication protocols and an electronic commerce protocol for various desired properties:

Logic	**Class of Protocols**	**Properties to Check**
BAN	authentication	authentication
AUTLOG	authentication	authentication, privacy
Kailar	electronic commerce	accountability

3 Formalizing Interpretation

The informal *idealization* step is a widely acknowledged limitation of BAN-like belief logics [Syv91, NS93, MB94]. Abadi and Needham provide a set of practical guidelines for constructing good idealizations [AN96]. These guidelines are unsuitable to formal verification, but furthermore they are more restrictive than necessary, so they are often "legitimately" violated in practice. This makes it hard to identify the truly improper idealizations that lead to problems.

The form of the BAN-like logics suggests a natural expression of idealizations: as extra rules added to the logic. Each of these rules can represent an "interpretation" that protocol participations can legitimately make. For instance, if Alice sees the signed message `[A.B.K]`, she might be allowed to assume that the meaning of that message is $A \stackrel{K}{\longleftrightarrow} B$. (We use the notation `[X.Y.Z]` to represent a concrete message with three fields.) This interpretation might correspond to a rule whose premise is $P\ \mathit{believes}\ Q\ \mathit{said}\ [P.Q.K]$ and whose conclusion is $P\ \mathit{believes}\ Q\ \mathit{said}\ P \stackrel{K}{\longleftrightarrow} Q$. With these extra interpretation rules, we can start from the initial assumptions and the concrete protocol messages, and derive all properties from there. We will allow interpretation rules in any of the following forms:

$$\frac{P\ \mathit{sees}\ X}{P\ \mathit{sees}\ Y} \qquad \frac{P\ \mathit{believes}\ Q\ \mathit{said}\ X}{P\ \mathit{believes}\ Q\ \mathit{said}\ Y} \qquad \frac{P\ \mathit{believes}\ Q\ \mathit{says}\ X}{P\ \mathit{believes}\ Q\ \mathit{says}\ Y}$$

Here, X is a pattern matching concrete messages, and Y, idealized messages. We will present further limits on the allowed interpretation rules later.

There are other approaches to bridging the "idealization gap." Since idealized protocols can exhibit different behaviors from their concrete counterparts, we might decide to abandon reasoning about beliefs and do all verification at the concrete level. This is what recent verification techniques based on model-checking do [Low96, MMS97, MCJ97]. While this approach has the appeal of producing counterexamples and not requiring the construction of idealizations, it does have some disadvantages. There is no longer any formal notion of the "meaning" of a message, and there is little indication of *why* a protocol works. We can analyze data flow: secrets successfully communicated, information leaked, and so on, and we can check the "correspondence" properties defined by Woo and Lam [WL93], but we cannot verify the richer and more intuitive belief properties. We cannot factor out the abstract "core" of a protocol from its various possible implementations.

Mao has proposed a method of developing idealizations in a principled way [Mao95], by breaking the idealization process down into a sequence of incremental steps, each of which can be justified individually. We choose a different approach that is suitable for automation via theory generation, and that we feel more directly expresses the desired qualities of idealizations.

4 Honesty

Burrows, Abadi, and Needham recommend, "for the sake of soundness, we always want to guarantee that each principal believes the formulas that he generates as messages" [BAN90]. This requirement has intuitive appeal: the protocol should not require principals to "lie." If participants were free to send arbitrary messages, recipients could derive faulty conclusions about the senders' beliefs from those messages. We will refer to this kind of restriction as an *honesty property*. In the next two sections, we discuss honesty first for idealized protocols, and then in the context of explicit interpretations.

4.1 Honesty in Idealized Protocols

We need a more precise statement of the honesty property; the goal is to prevent message recipients from deducing beliefs of other principals that are invalid. To achieve this, we must consider the circumstances under which one principal can decide another principal holds some belief. This typically happens through the application of a message-meaning rule, such as the rule for interpreting messages signed with a shared key, followed by the application of a nonce-verification rule. We focus on the message-meaning step: any message (formula) that is encrypted under some key (e.g., $\{A \overset{K_{ab}}{\longleftrightarrow} B\}_{K_s}$), or combined with some secret (e.g., $\langle A \overset{Nb}{\rightleftharpoons} B\rangle_{Na}$) could potentially be interpreted as a belief of the principal doing the encrypting or combining. Therefore, we will state the honesty property in terms of messages *signed*, where a principal signs a message whenever it applies encryption or secret-combination to that message. (Note that this notion is broader than that of public-key digital signatures.) A principal takes responsibility for any statement it signs, since those are the statements that might later be used by other principals to derive beliefs of the signer.

We can now state the honesty property:

> **Honesty property** (for idealized protocols): For every message component M that a principal P *signs*, it must be true that P $believes$ M at the point at which P sent the message containing M.

The requirement that the belief hold at the time the message is sent prevents circular situations in which two (or more) principals send each other statements that neither believes initially, but that both come to believe after receiving the other's message.

Most of the idealized protocols in the published BAN analyses will not pass this test, since they typically contain messages like $\{T_a, A \overset{K}{\longleftrightarrow} B\}_{K_{ab}}$, where the signer does not actually "believe" T_a. We can fix this by replacing each troublesome message fragment, X, with some formula the sender actually believes, such as $A\ sees\ X$ or $fresh(X)$. This approach requires introducing some extra "seeing" rules, such as, $P\ sees\ Q\ said\ X\ \vdash P\ sees\ X$.

4.2 Honesty with Explicit Interpretations

The honesty property is fairly simple in the context of reasoning about idealized protocols, but when we introduce interpretation rules, it becomes more interesting. Since concrete messages have no direct meaning, it no longer makes sense to talk about believing the contents of a message; we can only discuss believing some interpretation of a message. Roughly, we want to extend the honesty property to say that, for every message component a principal signs, that principal must believe all possible interpretations of that message component. We must, however, refine the notion of interpretation for this definition to be useful.

By making interpretations explicit, we expose the possibility that a single concrete message may be interpreted differently in different runs of the protocol. This would make the honesty property difficult to verify; we would prefer to confine our reasoning to a single run. By imposing certain restrictions on the interpretation rules, we can ensure that no additional interpretations are possible in other runs of the protocol. If we require that interpretation rules make no mention of "run variables," then their results on a given concrete message will be the same in all protocol runs. By "run variables" we mean variables in the protocol description that assume different values in different runs of the protocol. This would normally include principal names like A and B and specific keys and nonces. A trusted server name, S, would be considered a run variable unless the server's identity is fixed across all runs. This restriction, which corresponds loosely to Abadi and Needham's "explicitness" principle, is actually stronger than necessary. We are investigating ways to safely relax this restriction in order to accommodate more protocols.

Protocols involving shared secrets (not keys) can present a challenge. We choose not to provide a secret-combining operator ($\langle X \rangle_Y$) at the concrete level. Providing one would oblige the low-level implementation to distinguish secret-combining from ordinary concatenation. To allow more flexibility, we use interpretation rules to decide when a given concrete message is meant to convey a secret-combination. This means that for some messages, two levels of interpretation will be required: one to interpret a concrete message as a secret-combination, and a second to interpret the contents of the secret-combination.

We do model encryption explicitly at the concrete level, so there is no need for interpretations to introduce encryption. To preserve orthogonality with the message-meaning rules, interpretations should also not perform decryption. We can enforce this through a simple syntactic restriction that no encryption operators can appear in the premise or conclusion of an interpretation rule.

The honesty property for the explicit interpretations case is as follows:

> **Honesty property** (with explicit interpretations): For every message M that a principal P encrypts, and every formula I, where I is an interpretation of M from P, it must be the case that $P\ believes\ I$ at the point at which P sent M.

In this context, we say that message M from P has an interpretation I if there exists some Q such that we can derive $Q\ believes\ P\ said\ I$ or $Q\ believes\ P\ says\ I$ from $Q\ believes\ P\ says\ M$.

Together with the syntactic restrictions on interpretation rules, the honesty property provides many of the guarantees we would like to hold for idealizations.

5 Status and Future Work

We are developing a prototype verification environment, REVERE, which uses the theory-generation approach. It currently supports checking belief properties with a variety of logics, and honesty properties using a BAN variant similar in flavor to AUTLOG. Since the publication of our early performance results, we have implemented some optimizations that produced substantial speedups; typical verifications now take under one minute, rather than five to ten. This speed makes a quick design-and-debug cycle possible during protocol development.

Now that we can reason about concrete and idealized levels of protocol description together, we are in a position to capture a wider range of properties in a belief-logic context. The honesty property is just a first step. Honest principals can still cause damage if they are allowed to act in manners that are inconsistent with their beliefs; for instance, a principal may reveal shared secrets to untrusted third parties. We are currently investigating ways to apply theory-generation to checking *responsibility* properties, in order to expose this sort of misbehavior. Finally, we intend to check *feasibility* properties, which assert that principals have sufficient information at each step in a protocol to carry out their assigned roles.

We believe that these classes of properties complement each other in such a way that we will be able to make stronger claims about a protocol that satisfies all of them than we could using BAN-style reasoning alone. In addition, the implementations of these checks can be simplified by sharing inputs and results;

for instance, the feasibility check could produce as a side effect the set of signed message fragments which the honesty check requires.

References

[AN96] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[BAN90] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[Kai96] Rajashekar Kailar. Accountability in electronic commerce protocols. *IEEE Transactions on Software Engineering*, 22(5):313–328, May 1996.

[KW94] Volker Kessler and Gabriele Wedel. AUTLOG—an advanced logic of authentication. In *Proceedings of the Computer Security Foundations Workshop VII*, pages 90–99. IEEE Comput. Soc., June 1994.

[KW96] Darrell Kindred and Jeannette M. Wing. Fast, automatic checking of security protocols. In *Proc. of the USENIX 1996 Workshop on Electronic Commerce*, pages 41–52, November 1996.

[Low95] G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, November 1995.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055. Springer-Verlag, March 1996. Lecture Notes in Computer Science.

[Mao95] Wenbo Mao. An augmentation of BAN-like logics. In *Proceedings of the Eighth IEEE Computer Security Foundations Workshop*, pages 44–56, June 1995.

[MB94] Wenbo Mao and Colin Boyd. Development of authentication protocols: Some misconceptions and a new approach. In *Proceedings of the Seventh IEEE Computer Security Foundations Workshop*, 1994.

[MCJ97] W. Marrero, E. M. Clarke, and S. Jha. Model checking for security protocols. Technical Report CMU-CS-97-139, Carnegie Mellon University, May 1997.

[MMS97] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murphi. In *Proceedings of the IEEE Conference on Secuirty and Privacy*, pages 141–15, 1997.

[NS93] B. Neuman and S. Stubblebine. A note on the use of timestamps as nonces. *ACM Operating Systems Review*, 27:10–14, April 1993.

[Syv91] P. Syverson. The use of logic in the analysis of cryptographic protocols. In Teresa Lunt and John McLean, editors, *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.

[WL93] Thomas Y. C. Woo and Simon S. Lam. A semantic model for authentication protocols. In *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 178–194, May 1993.