Analyzing Content-Based Message Blocking with the SVO Logic

by

Nathan Wilcox

A thesis submitted to the

Faculty of the Undergraduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Bachelor of Science

Department of Computer Science

2006

This thesis entitled:
Analyzing Content-Based Message Blocking with the SVO Logic
written by Nathan Wilcox
has been approved for the Department of Computer Science

Prof. Dirk Grunwald

Prof. Doug Sicker

Prof. John Black

Date ____________________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Wilcox, Nathan (B.S., Computer Science)

Analyzing Content-Based Message Blocking with the SVO Logic

Thesis directed by Prof. Dirk Grunwald

We introduce and define the Censoring Firewall Problem (CFP) where two colluders attempt to transmit banned messages through a firewall. We analyze the problem with the SVO logic to prove conditions necessary for the colluders to succeed.

This is a novel application of SVO to a problem for which it was not originally designed. Our analysis illustrates shortcomings of SVO to our approach. Our primary contribution is the concept of a computable filter function which allows us to adapt SVO to the CFP.

Our contribution shows how SVO (a formal view) when applied to the CFP reduces to a problem of computability, highlighting the interface between the formal method and computational soundness perspectives.

Contents

Chapter

Contents

Chapter 1

Introduction

We study content-based censorship by a firewall using the SVO logic[1] . Our primary contribution is the concept of a computable filter function, which allows us to adapt SVO to apply to our problem definition. Using this approach, we show that it is possible for colluders to transmit banned messages if they agree on an obfuscation transformation based on the firewall's filtering policy.

In this introductory chapter we motivate and define the Censoring Firewall Problem (CFP). In Chapter 2 we firmly embed our analysis in a larger context of possible approaches to clarify our assumptions and the direction further research should take. Chapter 3 is our specific analysis with the SVO authentication logic, which begins with an illustrative analysis to motivate our main contribution. Our main contribution, the notion of a computable filter function which defines the firewall policy, is presented in Section 3.2.1. We proceed to refine the illustrative analysis using the concepts we develop. Chapter 4 presents avenues for future research, including gaps in our approach, broadening the CFP definition, extending SVO by integrating features from other logics, and analyzing further constraints on the filter function. We conclude in Chapter 5.

---

[1] The SVO logic is presented in [3] and [2], of which we mostly rely on the latter. The acronym is based on the authors' names, Syverson and van Oorschot.

1.1 Motivation

Traditionally firewalls are seen as improving the security of a group of computers against attacks from the "outside". In this view, the firewall is seen to represent the interests of a given protected host. An outside attacker wishes to send messages to the protected host which may compromise its security. The intended recipient and the firewall share the goal of blocking harmful messages.

Another scenario has become common in computer networks as the goals and relationships between human users and organizations grows more varied. The defining characteristic of this scenario is that the message recipient has **goals in conflict** with the censor. Because of this, the recipient may collude with the sender to circumvent the message blocking.

We use the term "censorship" to represent this conflict of goals. We stress our use of this term does not imply some common, prescriptive connotations of the term (namely that "censorship is bad"). For instance, the recipient may be a back-door running on a machine and upon receipt of the message, that host will initiate an internal network attack.

1.1.1 Errors in Policy

We also distinguish between goals, policies, and behavior. *Goals* are the object of human desires. *Policies* are computer-based implementations which attempt to achieve certain goals. *Behavior* is the actual result of a computer executing a specific policy in a particular circumstance.

One result of this research is to emphasize that although a given policy may appear to achieve a certain goal, it often does not. This is consistently an issue in security research, and serves as one of our primary motivations.

1.1.2 Extending Protocol Analysis

Techniques have developed for analyzing protocols which emphasize what participants can learn from messages, and how messages received influence knowledge and behavior. This is especially true in cryptography which assumes participants with conflicting goals.

The problem we wish to address is rooted in the transmission and knowledge of messages in the context of conflicting goals. However, the scenarios which interest us are sufficiently different from those historically considered by cryptographic analyses to require further research.

1.2 Problem Definition

Here we set out our assumptions and some definitions of the problem we wish to address. Specific definitions are given in Section 3.2.1 within the context of the analysis.

1.2.1 Network Model

1.2.1.1 Topology.

The network consists of principals related by a *connection topology* which constrains which principals may communicate directly. For our purposes, we consider a minimal case with only three principals, A, B, and C. The colluders, A and B, are each connected only to the censor, C. We use the convention that A is the message sender, and B the intended recipient.

This precludes analysis of route-based firewall policies, which is an important area of further research.

1.2.1.2 Messages.

All messages sent or received are either requests for forwarding (called *requests* for short) or the result of forwarding (*forwards* for short). A request contains a *destination* and a *body*. A forward contains only a body.

Each message belongs to a language specified in the SVO logic. We discuss this restriction more in Section 3.1.

1.2.2 Firewall Constraints

We limit the capabilities of the firewall in order to constrain the scope to what we consider the fundamentals of this problem. Changing any of these limitations would reveal new aspects of the censoring firewall problem, and we address each possible change in Section 4.3.

This set of capabilities is largely what differentiates this problem from typical scenarios addressed by cryptographic protocol logics, such as SVO. Typically, an entity called the adversary seeks to compromise some security goal according to some set of capabilities. Also, the adversary employs those capabilities arbitrarily to achieve compromise.

In our approach the term "the adversary" is misleading for several reasons. Each participant views any other participant with conflicting goals as an adversary, so there is no single objective adversary. Furthermore, the firewall follows a set, known policy, and does not arbitrarily employ its capabilities. Finally, although the colluders **do** arbitrarily employ their capabilities, they do not have perfect knowledge sharing and act independently[2] .

[2] Cryptographic protocol logics discussed in Section 2.1.2 consider multiple adversaries which may collude, in which case we might consider our colluders to be adversaries of this type.

1.2.2.1 Passive Blocking.

The firewall either forwards or blocks each received message once. The only messages it sends are forwards. All forwards are unmodified re-transmissions of a single request body. Every request triggers either a single forward or no action (when the request is blocked). By specifying the criterion for blocking messages, together with this constraint, we completely specify the firewall policy.

We call this passive because the firewall does not employ any other active attacks such as modifying request bodies before forwarding, initiating any messages aside from forwards, replaying any messages, etc. It differs somewhat from the concept of passive eavesdropper because the firewall intercepts messages and blocks them according to a set policy.

1.2.2.2 Message Independence.

Each message is considered individually, without regard to previously seen messages. We believe this to be common in practice.

1.2.2.3 Content-Based.

Criteria for filtering a message depends only on the body, not on its source or destination. We refer to this as *content-based blocking* as opposed to *route-based blocking*. We consider these to be orthogonal considerations for blocking criteria, and thus our analysis could be complemented by considering route-based blocking separately. Content-based firewalls are common in the wild, although perhaps less common than route-only firewalls.

1.2.3 Defining Censorship

Given the network model and firewall capabilities above, we now build up to a concise definition of censorship[3] :

Any message, M, belongs to a language, $\mathcal{M}_{\mathcal{T}}$, of primitive terms, $\mathcal{T}$. The primitive terms are a set of constant terms and the language is the closure of the primitive terms over certain operations[4] .

The set of *banned* messages, $\mathcal{B}$, is a subset of the language: $\mathcal{B} \subset \mathcal{M}_{\mathcal{T}}$. Likewise the *allowed* messages, $\mathcal{A}$, complements the banned messages: $\mathcal{A} \cup \mathcal{B} = \mathcal{M}_{\mathcal{T}}$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$.

Using these terms we can define censorship according to this policy: If C receives request $M \in \mathcal{A}$, then C sends the corresponding forward, else the request $M \in \mathcal{B}$ and C does nothing.

---

[3] Whenever our definition coincides with one in the SVO logic in [2], we use the same symbols for consistency.

[4] The specific primitives (such as keys and nonces) and construction operations (such as concatenation and encryption) are discussed more thoroughly in Section 3.1.2.

Chapter 2

Background

In this Chapter, we embed the SVO logic in the larger context of cryptographic protocol analysis techniques. We also briefly introduce SVO in detail, surveying those parts necessary in understanding our analysis.

The first section is an overview of formal methods used in cryptographic protocol analysis. This is followed by an introduction to belief logics and SVO in particular. The last section reviews two distinct approaches to cryptographic protocol analysis and work on relating them. This is relevant to our work which also relates these two perspectives.

2.1 Discrete Formal Methods

We summarize a variety of formal methods recently developed, relating them to our approach. This overview follows closely that of Meadows in [6]. These discrete methods view operations (such as encryption or message transmission) as atomic and deterministic. This contrasts with computational approaches which consider probabilistic cryptanalytic attacks.

2.1.1 Foundations

In [7], Dolev and Yao present models for analyzing protocol security. These models rest on assumptions which have become standard in formal methods. For the most part, we adhere to the same assumptions, but with important differences which we make explicit.

2.1.1.1 Perfect Public Key System.

The one-way (asymmetric) encryption functions are considered unbreakable, IE: invulnerable to computational attacks. We do not rely on this assumption because our results assume no cryptanalytic attacker.

2.1.1.2 Solved PKI.

The authors set out two assumptions which we group, namely that everyone knows all public keys, and that each principal has a private key known only to it. As we discuss below, successor analysis techniques allow more flexibility in these kinds of assumptions to cover a greater range of protocols. In other words, these assumptions can be expressed as explicit premises in our analysis and will be considered there.

2.1.1.3 Uniform Protocol.

The protocols in question apply to **any** set of principals who agree to employ it, by taking on requisite roles. That is, the protocols are not specific to individual principals. This is a standard assumption we also adopt. We prefer to rephrase it by saying that all protocol dependencies on the uniqueness of a participant are either due to topology or represented within the protocol as parameters (for example by possession of secrets).

2.1.1.4 Participants and Intruder Capabilities.

Perhaps the most influential aspect of [7] is the power attributed to the intruder (aka saboteur). Dolev and Yao distinguish between *passive* versus *active* eavesdroppers. They argue that the latter represent an important and (at the time) under-represented problem in security. They set out to model communications between only two participants, but which may be tampered with actively by an intruder. The intruder can intercept any message (possibly blocking propagation), can initiate the protocol as a participant, and can modify any message en route.

In addressing the CFP, we focus on a different aspect of communication, neglecting any intruder. The firewall is limited by the firewall constraints given in Section 1.2.2, and the colluders behavior is defined by those constraints.

As we discuss below the successors of Dolev and Yao relax the assumptions of the intruder capabilities in order to have more flexibility in modeling it. Dolev and Yaos' active versus passive distinction can be thought of as two extremes in a space of possible intruder capabilities. In this sense, we may view the firewall as a very limited intruder.

2.1.2 Overview of Formal Approaches

Meadows gives an overview of three classes of approach to protocol analysis, which we briefly summarize. Each of these approaches has successfully exposed previously unnoticed attacks.

State Exploration Techniques. This class of approach uses an automated tool to explore a state space specified by the analyst. Commonly specifications are based on a Dolev-Yao model. Particular states are defined as security violations, then the automated tool attempts to reach such attack states. If the tool does indeed reach such a state, it can provide the sequence of state transitions leading to that state. Inductive theorem proving complements such analyses, for example by proving the search space for the automated tool is too large for attacks to be discovered in computable time.

Type Checking. Meadows refers to this as "perhaps the newest approach" in the field, which assigns types to messages and channels, and represents security flaws as type violations. Type checking can be automated like state exploration, but certain classes of infinite systems can also be handled.

Belief Logics. A third class of analysis methods is belief logic, which includes the SVO logic. These methods represent relationships between principals, data, and their beliefs about that data as modal operators in a formal logic. Inference rules are used to prove security properties, and a lack of proof may indicate a security flaw.

The seminal work in this field is the BAN logic[1] given in [8], which inspired a variety of descendant logics, each addressing different concerns. In [1], Syverson gives an overview of the drawbacks of BAN, and reviews several successors each addressing different flaws of BAN. Syverson and van Oorschot developed the SVO logic to unify the advantages of these successors. We review SVO in the next section.

Another logic of note is proposed in [4], we refer to as GS.[2] GS addresses many of the drawbacks we expose in SVO, but unfortunately it does not accommodate arbitrary computable functions, which are essential for our results. We discuss this more fully in Section 4.4.

2.1.3 Review of SVO

We now review the most relevant parts of SVO for our analysis.

2.1.3.1 The Message Language.

The language of messages in SVO, $\mathcal{M}_{\mathcal{T}}$, was presented in Section 1.2.3. Specifically it is the closure of a set of primitive terms, $\mathcal{T}$, and formulae, over all functions $F(X_1, \ldots, X_n)$. Such functions include concatenation: $F(X, Y) = (X, Y)$, and encryption: $F(M, k) = \{M\}_k$.

[1] The acronym "BAN" comes from the authors, Burrows, Abadi, and Needham. This convention has stuck in the field, with each logic named after the authors.

[2] This follows the convention in naming these logics.

2.1.3.2 Inference Rules.

Modus Ponens and *Necessitation* are the only two inference rules in SVO. If φ and ψ represent any formulae, then we have these definitions:

Modus Ponens: From $\varphi \supset \psi$ and φ infer ψ

Necessitation: If φ follows from the axioms, infer P *believes* φ

2.1.3.3 Axioms.

Here we describe those SVO axioms (and axiom schemata) used in our derivations. Note, these axiom labels come from [2], which differ from an earlier version of SVO given in [3].

The following axiom ensures that P believes everything which logically follows from its beliefs. Let φ and ψ be any formulae, then:

Ax1. $P\ \textit{believes}\ \varphi \wedge P\ \textit{believes}\ (\varphi \supset \psi) \supset P\ \textit{believes}\ \psi$

If P receives an encrypted message and has access to the key, then P receives the plaintext:

Ax8. $(P\ \textit{received}\ \{X\}_k \wedge P\ \textit{sees}\ \tilde{K}) \supset P\ \textit{received}\ X$

Anything received is seen:

Ax10. $P\ \textit{received}\ X \supset P\ \textit{sees}\ X$

P can see any function of anything it sees:

Ax12. $(P\ \textit{sees}\ X_1 \wedge \ldots \wedge P\ \textit{sees}\ X_n) \supset (P\ \textit{sees}\ F(X_1, \ldots, X_n))$

2.2 Computational Versus Formal Perspectives

According to Abadi and Rogaway in [5] there are two views of cryptography that have developed, treating the subject differently. They present an equivalence between definitions of security given from each perspective as a first step at bridging the gap. We give a brief comparison here, and justify why we choose a formal method approach.

2.2.0.4 Computational View.

In the computational view, cryptographic operations are considered algorithms operating on strings of bits. Security properties are defined taking probability and computational cost into account.

2.2.0.5 Formal View.

This view represents cryptographic operations symbolically in a formal language. The relevant properties of a cryptographic operation are assumed as part of the formal semantics. **Given** that those properties hold, formal methods allow analysis of more complex systems built on top of such primitives.

Chapter 3

Analysis

In this section we present two contrastive analyses of the CFP using the SVO logic. The first analysis follows a typical SVO procedure. It is illustrative of drawbacks to this typical approach. The second analysis presents a refined approach which more accurately describes the CFP. Between these analyses, we present key concepts necessary for the refined analysis. The essential concept is the main contribution of our work, the *computable filter policy.*

3.1 Illustrative Analysis

This illustrative analysis uses SVO to model a specific instance of the CFP in which the colluders obfuscate a banned message with encryption. It proceeds in the manner typical of examples given in [2]. Along the way we highlight shortcomings of this approach.

3.1.1 Goals

The colluders wish to send a banned message, $M \in \mathcal{B}$. We define expressions which represent the colluders achieving their goal. If we can derive these goal expressions from some starting assumptions, then those assumptions are sufficient for the colluders to realize their goal. If not, the censor's goal is realized.

For simplicity, we consider A and B to have the same two goals: B must receive the banned message and B must properly interpret this as the case. The corresponding SVO expressions are:

G1. B *sees* M

G2. B *believes* B *sees* M

The firewall should be content if either of these cannot be achieved. If B receives the message but cannot realize this fact (IE: G1 but not G2), it cannot act on the contents. On the other hand, if B believes it has received the message when it has not, it will act mistakenly on the contents.

Notice that these goals don't include any statement about A believing B has received the message. This means A cannot know if the goal is reached.

3.1.2 Protocol

We present a minimalist protocol specification, which leaves out routing information[1] and only contains the main content. We follow the specification with discussion of two shortcomings of SVO protocol specifications.

M1. $A \rightarrow C : \{M\}_k$

M2. $C \rightarrow B : \{M\}_k$

3.1.2.1 Invariable Message Constraint.

The specification for M1 is misleading, because it implies the protocol is "a client sends an encrypted message to the firewall". We would rather express "a client sends an arbitrary message in a given language to the firewall". We cannot specify a protocol with arbitrary messages, only messages fitting a prescribed form.

[1] Recall that our network topology implies a single route. A sends a request by convention, and B is the only possible destination.

3.1.2.2 Non-branching Protocol Constraint.

SVO does not handle branching protocols in which different messages may be sent depending on previous messages. The CFP exemplifies such a branch: Depending on the request, either a forward is sent, or nothing is sent. We must specify only one of these cases, which places constraints on the request.

In this analysis we assume the firewall forwards the message. We refer to this as the *forwarding assumption*. The firewall only forwards messages in the allowed set, so it follows that $\{M\}_k \in \mathcal{A}$.

3.1.3 Initial State

These premises define our assumptions about the principals' states before the protocol begins, including which terms they have access to (using the *sees* syntax), and which beliefs they hold.

The first premise expresses that B has the symmetric key which encrypts M:

P1. B *sees* k

The following premise is necessary to derive G2. If B does not believe that it has the secret key, k, then B cannot believe it can perform decryption.

P2. B *believes* B *sees* k

3.1.4 Messages

This subsection presents the assumptions about message reception, comprehension, and interpretation.

3.1.4.1 Message Reception.

As is standard in SVO analyses, we must assume the principals receive the messages specified in the protocol, or else we cannot make any claims about the results of such receptions:

P3. C *received* $\{M\}_k$

P4. B *received* $\{M\}_k$

3.1.4.2 Message Comprehension.

Unlike the predecessor BAN logic, given in [8], SVO forces analysts to be explicit about which fields in a message are comprehended by a recipient. Comprehended fields must either contain sufficient redundancy or the recipient must have the proper expectation of the field value, in order for the recipient to act directly on the contents.[2]

The request, a result of encryption, is opaque to C. Because of this we assume C comprehends the message to be an unrecognized string, represented as the primitive $*_1$.[3]

P5. C *believes* C *received* $*_1$

We also assume B cannot directly comprehend the payload (which is encrypted), but must assume its contents to decrypt it. In other words B knows it received an opaque payload, but it must have other beliefs (given next) about that payload in order to act upon it.

P6. B *believes* B *received* $*_2$

[2] Note, a principal may still have expectations about the contents of an uncomprehended fragment (such as "this contains an message encrypted with key k" or "this contains a message intended for another principal" or "this represents a nonce".). Such expectations are given as separate interpretation assumptions in SVO.

[3] The primitives $*_i$ are reserved for this purpose in SVO. The subscript i allows representing distinct unrecognized fragments.

3.1.4.3 Message Interpretation.

Comprehension is the first of two stages SVO introduces to reduce the ambiguity present in the BAN idealization process. In this stage we assert how principals interpret message fragments that are not comprehended. Specifically we assume when B receives the opaque encrypted message, it interprets[4] that to be some message encrypted with the secret key k:

P7. B *believes* (B *received* $*_2 \supset B$ *received* $\{M\}_k$)

3.1.4.4 Belief Conjugation.

If a principal holds two beliefs, it seems trivial to suppose it also holds a belief about the conjugation of both objects of those beliefs. In symbols: (P *believes* $\varphi \wedge$ P *believes* $\psi) \supset P$ *believes* $(\varphi \wedge \psi)$. Surprisingly, SVO appears to lack such an axiom, so we explicitly assume a specific case as a premise:

P8. (B *believes* B *received* $\{M\}_k \wedge B$ *believes* B *sees* $k) \supset$

B *believes* (B *received* $\{M\}_k \wedge B$ *sees* k)

3.1.5 Derivations

With the above assumptions, we derive the two goals of the colluders, A and B. The first two derivations reach G1 because B received the encrypted message and had the appropriate decryption key, and thus sees the message:

D1. B *received* M

MP of P4, P1 applied to Ax8.

D2. B *sees* M

MP of D1 applied to Ax10.

[4] Without this assumed interpretation, B would have no justification for decrypting with key k.

The next four derivations are more subtle, to reach the subtler goal, G2. First, B believes B received an encrypted message because it believed it received an incomprehensible message and it believed that to be interpreted as a message encrypted with k:

D3. B *believes* B *received* $\{M\}_k$

MP of P6, P7 applied to Ax1.

Now, B believes decryption of the message $\{M\}_k$ yields the M. This belief is G2.

D4. B *believes* $[(B \textit{ received } \{M\}_k \wedge B \textit{ sees } k) \supset B \textit{ received } M]$

Nec. of Ax8

D5. B *believes* $(B \textit{ received } \{M\}_k \wedge B \textit{ sees } k)$

MP of D3, P2 applied to P8.

D6. B *believes* B *received* M

MP of D4, D5 applied to Ax1.

3.1.6 Summary

In this derivation we show that the colluders can transmit a banned message through the firewall given two conditions: The recipient has the decryption key and knows it, and more importantly, the firewall forwards the encrypted message.

The latter is not clearly represented in the analysis, aside from the ad hoc forwarding assumption stated in Section 3.1.2. That assumption complements our assumptions $\{M\}_k \in \mathcal{A}$, but there is no direct relation between these assumptions. In the next section we propose definitions which rigorously establish such a relationship.

3.2 The Computable Filter Policy

We now introduce mechanisms within SVO to explicitly relate the forwarding assumption of Section 3.1.2 to our concept of the sets $\mathcal{A}$ and $\mathcal{B}$. This rigorously reduces the problem to one of computability, demonstrating the interface between formal methods and computational soundness methods.

3.2.1 The Filter Function

Previously we constrained the banned messages only by $\mathcal{B} \subset \mathcal{M}_{\mathcal{T}}$. Not only is this vague but it omits consideration of an important notion: How does C determine if a message belongs to this set?

The firewall must have some computable criterion by which it chooses which messages to block, which we call the *filter*, $\mathcal{F}$. The filter defines the firewall policy because of the firewall's passive blocking constraint.

The firewall constraints given in Section 1.2.2 imply the following filter properties:

Computability. The firewall acts by its passive constraint according to the filter criterion, so $\mathcal{F}$ must be computable.

Decidable. The filter must decide which set $\mathcal{A}$ or $\mathcal{B}$ a message belongs to within pragmatic time bounds. This property constrains the range of $\mathcal{F}$ to two outcomes. We use the term decidable to emphasize $\mathcal{B}$ must be a decidable language.

Stateless and Content-Based. The filter decision depends only on a single input message, and no other context, due to message independence. This defines the domain of $\mathcal{F}$ to the set of all possible messages, $\mathcal{M}_{\mathcal{T}}$.

3.2.1.1 Domain and Range.

We introduce a set of *filter decisions*, denoted $\mathcal{D}$, containing only two special primitives, *Allowed* and *Banned*. We require the filter to map all messages to a filter decision, so: $\mathcal{F} : \mathcal{M}_{\mathcal{T}} \mapsto \mathcal{D}$.

The banned set is now defined[5] in terms of the filter, $\mathcal{B} \equiv \{M : \mathcal{F}(M) = \mathit{Banned}\}$. With this definition, we say a message M *passes* the filter if $\mathcal{F}(M) = \mathit{Allowed}$.

3.2.2 Generalized Message Obfuscation

With a refined notion of the firewall policy, we now generalize the message transformations by which the colluders can achieve their goals. Intuitively, the sender translates a banned message to an allowed message. The allowed message is forwarded, and the recipient (who must know the reverse transformation) can recover the original. In so generalizing, we must relax an assumption implied by encryption; a generally obfuscated message may appear as another meaningful message, whereas an encrypted message is always assumed to be opaque.

In the illustrative analysis of Section 3.1, we assume $\{M\}_k \in \mathcal{A}$. However, nothing requires this to be true, and we may just as well assume $\{M\}_k \in \mathcal{B}$, with caveats.[6] Instead of considering only the specific encryption message transformation, $\{M\}_k$, we generalize the message transformation, using the filter properties as a basis.

3.2.2.1 Computations to Obscure and Reveal.

We assume both $\mathcal{M}_{\mathcal{T}}$ and $\mathcal{B}$ are countably infinite, and therefore countably infinite computable one-to-one mappings between them exist, which we call *obfuscation transformations*. Let an obfuscation function, $G : \mathcal{M}_{\mathcal{T}} \mapsto \mathcal{A}$, denote one such mapping from all messages to only allowed messages, and let the corresponding *revelation* function, G^{-1}, be its inverse. For any message M we have $G(M) \in \mathcal{A}$ and also $G^{-1}(G(M)) = M$, both by definition.

Mapping Agreement. The goal of the colluders can be achieved by using any such pair of transformations. However, they must agree on which specific transformation

[5] The allowed set is defined in the analogous manner.
[6] There is an important caveat related to computability here, which we address in Section 4.2.

to employ. We assume a unique one-to-one mapping from the natural numbers to each obfuscation transformation pair is available to the colluders.

We denote the j-th such obfuscation function as G_j and the corresponding revelation function as G_j^{-1}. The colluders can agree on which pair of transformations to use by agreeing on an index j. We also require a principal to know j in order to compute G_j or G_j^{-1}.[7]

Comparison to Symmetric Encryption. Although the properties of these transformations are purposefully similar to those of symmetric encryption, there are important differences. The obfuscation transformations are defined in terms of the filter function. If the filter function is quite simple, then so might the obfuscation transformations (so they may be vulnerable to even simple cryptanalysis).

Another important distinction is that an obfuscated message may appear as another meaningful message, whereas typically symmetrically encrypted messages are assumed to be opaque. This important distinction is discussed below in Section 3.3.5.2 when considering message comprehension.

To further clarify, **if** the ciphertext is allowed, $\{M\}_k \in \mathcal{A}$, **then** symmetric encryption transformations are a subset of the obfuscation transformations.

3.3 Refined Analysis

The definitions of the computable filter and obfuscation transformations pave the way for the following refined analysis.

3.3.1 Goals

We derive the same goals as in the illustrative analysis.

G1. *B sees M*

[7] For the purposes of SVO, we consider $G_j(M)$ to be a function of both M **and** j.

G2. *B believes B sees M*

3.3.2 Protocol

The protocol closely resembles that of the illustrative analysis in Section 3.1, except we have replaced the notation for standard encryption with an obfuscation function on M. This still carries the implicit assumption, due to M2, that the firewall does not block the message. However, in this analysis we derive that the firewall sends M2 rather than merely assuming it.

M1. $A \rightarrow C : G_j(M)$

M2. $C \rightarrow B : G_j(M)$

3.3.3 Filter-Related Premises

Before presenting protocol related premises, we present three premises related to the filter and obfuscation.

3.3.3.1 Filter Decisions.

Recall that filter decision primitives are treated specially in that we assume the firewall does not begin the protocol run with access to them. This constraint makes the filter function the only channel by which the firewall can gain access to these decisions.

This gives us our first premise[8] with regards to the filter function. By definition $\mathcal{F}(G_j(M)) = \textit{Allowed}$, so seeing either implies seeing the other. If C sees the filter results of an obfuscated message, C sees the *Allowed* decision.

P1. $C \textit{ sees } \mathcal{F}(G_j(M)) \supset C \textit{ sees Allowed}$

[8] This premise may not be logically necessary, but we include it to make the derivation semantics more explicit.

3.3.3.2 Firewall Policy.

In the illustrative analysis, we merely assumed the firewall forwards the request due to the complementary interpretation assumption that the firewall does not believe the encrypted request to contain a banned message.

However, with our concept of filter decisions defined, we can now express the firewall policy as a premise. If the firewall sees a message, and sees the *Allowed* filter decision, then the firewall sends the message.

P2. $[(C\ sees\ G_j(M)) \wedge (C\ sees\ Allowed)] \supset C\ says\ G_j(M)$

3.3.3.3 Implied Routing.

Because SVO is intended for protocols with small finite numbers of participants, it lacks message syntax for routing. In the simplified network topography of the CFP, we are safe to assume anything C says is received by B.

P3. $C\ says\ G_j(M) \supset B\ received\ G_j(M)$

3.3.4 Initial State

The colluder goals G1 & G2 do not require prevention of eavesdropping, because the firewall is constrained by the Passive Blocking policy. Therefore, the transformation index, j, need not be secret. The colluders only need to agree on an obfuscation transformation pair.[9]

P4. $B\ sees\ j$

We also require B to have a belief in seeing j to derive G2, in an analogous manner to the illustrative analysis.

P5. $B\ believes\ B\ sees\ j$

[9] We omit statements about A agreeing on j which is implicit in M1.

3.3.5 Messages

Message reception, comprehension, and interpretation is greatly changed from the illustrative analysis. These differences demonstrate the effects of our filter function mechanism.

3.3.5.1 Message Reception.

We only take message reception of the first message as a premise. Reception of the second message follows from the filter decision, firewall policy, and implied routing.

P6. C *received* $G_j(M)$

3.3.5.2 Message Comprehension.

The premises about comprehension diverge greatly from the illustrative analysis and standard SVO comprehension premises. The reason is that obfuscation and revelation transform one valid message into another (which need not be an opaque primitive, $*_i$). The principals will comprehend a transformed message as it appears. There is no way to determine whether it is an obfuscation result by comprehension.

In the illustrative analysis, opaque messages were comprehended as $*_i$ primitives. There is nothing to prevent the obfuscation transformation from mapping to a message other than a $*_i$ primitive. We consider this a more general view of comprehension, but there may be semantic caveats.[10]

For these premises, we let $G_j(M) = Y$, where $Y \in \mathcal{A}$ is any valid message (including both $*_i$ and other messages).

P7. C *believes* C *received* Y

[10] At this time we are uncertain about the soundness implications of generalizing comprehension in this way.

As mentioned under Message Reception, we choose to derive the transmission of M2, rather than assume it as a premise. This also means we must derive B's comprehension of the message from its reception, rather than assume it. To accomplish this, we take as a premise that if B receives the forward, B comprehends it as message Y.

P8. $(B\ received\ G_j(M)) \supset (B\ believes\ B\ received\ Y)$

3.3.5.3 Message Interpretation.

In traditional SVO message interpretation, premises are presented in which a principal interprets an opaque primitive, $*_i$, as another message. We allow more general interpretation in keeping with our more general comprehension mechanism.

The firewall comprehends the request as Y, and has no other interpretation of this. On the other hand, B interprets messages to represent something other than it comprehends them to be. This is a key ingredient in our concept of collusion and obfuscation. When B receives a message it interprets that to be an obfuscation of a different message, regardless of how it is comprehended. So we have:

P9. $B\ believes\ (B\ received\ Y \supset B\ received\ G_j(M))$

3.3.5.4 Belief Conjugation.

We again encounter the problem of belief conjugation mentioned in the illustrative analysis of Section 3.1. We solve it with the same approach, by taking as a premise a specific instance[11] of what we consider should be axiomatic[12] .

P9. $(B\ believes\ B\ sees\ G_j(M) \wedge B\ believes\ B\ sees\ j) \supset$

$B\ believes\ (B\ sees\ G_j(M) \wedge B\ sees\ j)$

[11] Notice how similar the two instances are: they both deal with beliefs about transformed messages and the appropriate parameter for the reverse transformation (whether encryption in the illustrative derivation, or obfuscation in this derivation).

[12] The axiom schemata we propose is $(P\ believes\ \varphi \wedge P\ believes\ \psi) \supset P\ believes\ (\varphi \wedge \psi)$

3.3.6 Derivations

With our refined premises laid out, we proceed with the derivation to reach the colluder goals G1 & G2 . In the first stage we prove the firewall forwards the request.

3.3.6.1 Passing the Filter.

We derive that the request passes the filter because C sees the request, and can apply the filter to see the result, which is *Allowed*.

D1. C *sees* $G_j(M)$

MP of P6 applied to Ax10.

D2. C *sees* $\mathcal{F}(G_j(M))$

MP of D1 applied to Ax12.

D3. C *sees Allowed*

MP of D2 applied to P1.

3.3.6.2 Forwarding the Request.

The message passes the filter and gets forwarded by the policy premise, P3. The forward is received by B according to the implied routing premise, P4.

D4. C *says* $G_j(M)$

MP of D1, D3 applied to P2.

D5. B *received* $G_j(M)$

MP of D4 applied to P3.

3.3.6.3 Revelation.

When B receives the forward and performs the appropriate revelation transformation, B achieves G1.

Receiving implies B has access to the revealed message via transformation. Here we consider G^{-1} to be a function of the index, j, and the obfuscated message, $G_j(M)$. The revealed message is just M and seeing the former is seeing the latter, by P2.

D6. *B sees M*

MP of D5 applied to Ax10.

3.3.6.4 Comprehension and Interpretation.

In receiving the forward, B comprehends it to be Y (as discussed above), and in turn interprets it to be the obfuscated message, $G_j(M)$. This interpretation is at the heart of the collusion we model in this report.

D7. *B believes B received Y*

MP of D5 applied to P8.

D8. *B believes B received $G_j(M)$*

MP of D7, P9 applied to Ax1.

3.3.6.5 Belief in the Revelation.

By Necessitation, B believes it sees the message interpretation:

D9. *B believes (B received $G_j(M) \supset B$ sees $G_j(M)$)*

Nec. of Ax10

D10. *B believes B sees $G_j(M)$*

MP of D8, D9 applied to Ax1.

Furthermore, B believes that if it sees an obfuscated message, $G_j(M)$, and the appropriate revelation parameter, j, then it can recover the message. (This comes also by Necessitation.)

D11. B *believes* $[(B \textit{ sees } G_j(M) \wedge B \textit{ sees } j) \supset B \textit{ sees } M]$

Nec. of Ax12

Finally, by the belief conjunction premise, B believes it can reveal the message M.

D12. B *believes* B *sees* $G_j(M) \wedge B$ *sees* j

MP of D10, P5 applied to P10.

D13. B *believes* B *sees* M

MP of D11, D12 applied to Ax1.

3.3.7 Summary

We apply the concept of the filter function and the obfuscation transformations to refine our analysis of the CFP.

The decision, policy, and routing premises allow us to derive the transmission of the forward, M2, rather than assume it. These premises are somewhat unwieldy because they require special treatment of the decision primitives, $\mathcal{D}$. More importantly, a careful reader will notice the conflict in verb tense in the routing premise, P3. This may represent a semantic flaw in our approach.

The obfuscation functions may transform messages into other non-opaque messages, and this may have soundness implications of which we are not aware. The flavor of interpretation is altered by this obfuscation property (which we reflect by using the symbol Y rather than $*_i$). We believe this emphasizes an important difference in comprehension and interpretation: the former may be thought of as an algorithmic process such as parsing, the latter captures assumptions of the intent of the parties involved.

This concludes our chapter on analysis, and we proceed next to overview open topics for further research.

Chapter 4

Further Work

Much remains open in understanding the CFP, and the relationship between protocol analysis logics and computability. This section is an overview of open topics.

4.1 Gaps in Analysis

The most critical path for future research is to address potential gaps in our analysis, which we review here:

4.1.1 Soundness

Some of our deviations from standard SVO analyses may rely on implicit deviations to the semantics of SVO. Those semantic deviations may affect the soundness of the changed logic. Those deviations are as follows.

4.1.1.1 Comprehending Obfuscated Messages.

Perhaps the most significant change we introduce is transformations between meaningful, comprehensible messages. In standard SVO, transformations such as $\{M\}_k$ do not result in "collisions of comprehension". In other words, no principal will mistakenly comprehend $\{M\}_k$ as Y. Instead they will always comprehend such transformations as opaque fragments, $*_i$.

If we allow comprehension of $\{M\}_k$ as a different message Y, then what prevents comprehending **any** unrecognized fragment as another message? Perhaps a more poignant example of this concern is comprehending a nonce, N_i, as a valid message, X.

In every SVO example we've seen, message transformations yield unique messages, and unrecognized fragments are always comprehended as opaque $*_i$ primitives. We deviate from this practice.

4.1.1.2 Implied Routing Premise.

In Section 3.3.3 we assume a premise of the form $C\ says\ G_j(M) \supset B\ received\ G_j(M)$. The verb tense of these connectives do not agree, and this may indicate an unexpected semantic result. If that result is not the intuitive one we attempt to express, then changing the semantics to fit our intuition may break soundness (or other important semantic results).

4.2 Open Computational Issues

4.2.1 Rigorously Defined Filter Functions

We introduce the defining characteristics of filter functions in Section 3.2.1, but neglect to go into more detail. This glosses over entire fields of literature on parsing, pattern recognition, intrusion detection, steganography, and cryptanalysis, to name a few.

Of particular interest is a cryptanalytic question: Can the set of banned messages **practically** include encryptions of a core set of banned messages. In the illustrative analysis of Section 3.1 we assumed $\{M\}_k \in \mathcal{A}$. Suppose we wish to make the inverse assumption, $\{M\}_k \in \mathcal{B}$. Can we define a filter that detects $\mathcal{F}(\{M\}_k) = Banned$ given that $M \in \mathcal{B}$ but k is unknown?

The answer to this question probably has many applications, but as we show in the case of the CFP obfuscation transformations exist as long as the filter meets the specified constraints. Even if $\{M\}_k \in \mathcal{B}$, we can obfuscate it to another allowed message.

4.2.2 Obfuscation Based on Partial Information

We assume the obfuscation transformations were defined in terms of the filter. But if the colluders only have partial information about $\mathcal{F}$, can they still agree on obfuscations that provably reach their goals?

If computational constraints further than we have specified can be proved for $\mathcal{F}$, then perhaps there are obfuscations which would work against any practical filter. This is perhaps a goal of cryptanalysis: Given an encryption (aka obfuscation) function, is it possible to determine whether a bitstring is within its range with only partial information about the plaintext or key?

4.3 Broadening the CFP Definition

Each of the firewall constraints in Section 1.2.2 could be relaxed in a search for more general results. We review each constraint and discuss the effects of relaxing it.

4.3.1 Passive Blocking.

A firewall which can spontaneously generate messages might launch "attacks" aimed at exposing two colluders.

For example, imagine the colluders are participating in an *embedded protocol* defined by a sequence of messages M_i, all of which are banned. A obfuscates an embedded protocol message, M_1, then sends it across the firewall. B receives the obfuscated message, reveals it, and generates M_2 according to the embedded protocol. Then B obfuscates M_2 and sends it back. This continues to the completion of a protocol.

In this scenario an firewall which is not constrained to passive blocking might try things like replaying obfuscated messages to learn more about the obfuscation transformation. For example, if the embedded protocol exchanges information about the obfuscation transformation, then this problem begins to look much like one of authentication in the face of an active intruder.

4.3.2 Message Independence.

As briefly mentioned in the last scenario, a firewall may try to record relationships between different requests to learn more information about the colluders.

4.3.3 Content-Based Filtering.

A firewall may also block messages based on route. Analyzing this would require allowing complicated network topologies. Colluders could attempt to bypass route-based filtering by forwarding messages around unblocked clients.

4.4 Synthesizing SVO and GS

Both analyses with SVO in Chapter 3 reveal certain limitations in application to the CFP. The GS logic, mentioned in Section 2.1.2, addresses these issues, but it does not address arbitrary computations which are essential to our results. The features of GS not found in SVO are that it addresses protocols which are open-ended both in participant topology and branching.

4.4.1 Open-Ended Participant Topology

GS can analyze protocols with an open-ended number of participants related by different topologies. This would allow broadening the CFP specification to include route-based censorship and collusion strategies.

4.4.2 Branching Protocols

We attempt to model a branching protocol in SVO, but our analysis is somewhat unwieldy. The GS logic addresses this issue by a *knowledge program* concept. A knowledge program defines how a principal reacts to receiving a message, which may include sending a new message. Axioms then support the notion that if a principal in a given state is running a given knowledge program and receives a given messages, then it sends another message. This is precisely what we wish to capture

Chapter 5

Conclusion

We introduce and define the CFP, then give two analyses in the SVO in search of solutions. The first analysis is illustrative of the drawbacks of SVO in addressing this problem. We introduce and define the computable filter function and related obfuscation transformations which address these drawbacks. Our second analysis applies these concepts to demonstrate their utility.

Our contribution shows how SVO (a formal view) when applied to the CFP reduces to a problem of computability, highlighting the interface between the formal method and computational soundness perspectives.

In doing so we show how colluders may always bypass message-based censorship if they agree on an obfuscation mapping derived from the firewall policy.

Bibliography

[1] Paul Syverson and Iliano Cervesato. The Logic of Authentication Protocols. *First International School on Foundations of Security Analysis and Design (FOSAD)*, 2000.

[2] Paul Syverson and Paul van Oorschot. A Unified Cryptographic Protocol Logic. *NRL CHACS Report*, 1996.

[3] Paul Syverson and Paul van Oorschot. On Unifying Some Cryptographic Protocol Logics. *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 1994.

[4] Paul F. Syverson and Stuart G. Stubblebine. Group Principals and the Formalization of Anonymity. *Formal Methods, VOL 1*, Sep, 1999.

[5] M. Abadi, P. Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). *IFIP TCS2000*, August 2000.

[6] Catherine Meadows. Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends. *Fromal Methods Section, Naval Research Laboratory*, 2003.

[7] On the Security of Public Key Protocols. D. Dolev and A. Yao. *IEEE Transactions on Information Theory*, March 1983.

[8] M. Burrows, M. Abadi, R. Needham. A Logic of Authentication. *Digital Equipment Corporation, University of Cambridge Computer Laboratoy*, ACM Transactions on Computer Science. February 1990