A Formalization of Off-Line Guessing for Security Protocol Analysis*

Paul Hankes Drielsma, Sebastian Mödersheim, and Luca Viganò

Information Security Group, ETH Zurich, CH-8092 Zurich, Switzerland
http://www.infsec.ethz.ch/~[drielsma|moedersheim|vigano]

Abstract. Guessing, or dictionary, attacks arise when an intruder exploits the fact that certain data like passwords may have low entropy, i.e. stem from a small set of values. In the case of off-line guessing, in particular, the intruder may employ guessed values to analyze the messages he has observed. Previous attempts at formalizing off-line guessing consist of extending a Dolev-Yao-style intruder model with inference rules to capture the additional capabilities of the intruder concerning guessable messages. While it is easy to convince oneself that the proposed rules are correct, in the sense that an intruder can actually perform such "guessing steps", it is difficult to see whether such a system of inference rules is complete in the sense that it captures all the kinds of attacks that we would intuitively call "guessing attacks". Moreover, the proposed systems are specialized to particular sets of cryptographic primitives and intruder capabilities. As a consequence, these systems are helpful to discover some off-line guessing attacks but are not fully appropriate for formalizing what off-line guessing precisely means and verifying that a given protocol is not vulnerable to such guessing attacks.
In this paper, we give a formalization of off-line guessing by defining a deduction system that is uniform and general in that it is independent of the overall protocol model and of the details of the considered intruder model, i.e. cryptographic primitives, algebraic properties, and intruder capabilities.

1 Introduction

Motivation. A serious vulnerability of security protocols lies in the simple fact that users tend to choose poor passwords, that is, passwords that are easy to remember and accordingly easy to guess (English words, for instance). This can give rise to *guessing* (or *dictionary*) *attacks* in which an intruder is able to guess a password or some other guessable data, and then somehow verify that his guess is correct [15, 19].

One usually distinguishes between *on-line* and *off-line* guessing. One speaks of on-line guessing when the intruder employs guesses when interacting with

* This work was partially supported by the FET Open Project IST-2001-39252 and the BBW Project 02.0431, "AVISPA: Automated Validation of Internet Security Protocols and Applications".

other agents, e.g. trying to log into a system using a guessed password. One speaks of off-line guessing when the intruder (who is not necessarily passive) employs guesses when analyzing observed messages, e.g. when trying to decrypt a message that is encrypted with a guessable password.

A number of approaches have been proposed to perform formal protocol analysis in the presence of on-line guessing, e.g. [9]. In this paper, we focus on off-line guessing. Several protocols, like the ones of the EKE family [7, 8], have been devised in an attempt to provide resilience to off-line guessing attacks. The idea behind these protocols is that, even if the intruder guesses the password correctly, he has no means of verifying whether his guess is correct or not.

In order to prove formally that such a protocol is correct, or detect attacks on it, it is necessary to have a precise definition of off-line guessing and of the potential vulnerabilities of protocols to such guessing attacks. A number of approaches, e.g. [10–13, 17], have been proposed to formally analyze protocols under the assumption that the intruder can perform off-line guessing.

The motivation underlying our work is based on the observation that these approaches suffer from two problems. First, all these different approaches are based on extending a Dolev-Yao-style intruder [14] with sets of inference rules to capture the additional capabilities of the intruder in the context of guessable messages. While it is usually easy to convince oneself that the proposed rules are correct, in the sense that an intruder can actually perform such "guessing steps", it is difficult to see whether such a system of inference rules is complete in the sense that it captures all the kinds of attacks that we would intuitively call "guessing attacks". Second, the proposed deduction systems are specialized to particular intruder capabilities and to particular sets of cryptographic primitives and operators (and, in some cases, e.g. [12, 17], they are even restricted to considering single guesses).

As a consequence, these systems are helpful to discover some off-line guessing attacks, but are not fully appropriate for formalizing what off-line guessing precisely means and proving that a particular protocol is not vulnerable to such guessing attacks.

Contributions. We propose, however, that there is a fairly simple intuition underlying off-line guessing, which captures a generic class of off-line guessing attacks without restriction to a particular intruder model or a set of cryptographic operations and their associated properties. In this paper, we overcome the above problems by precisely formulating this intuition to give a formalization of off-line guessing. More specifically, we define a deduction system that is uniform and general in that it is independent of the details of the considered intruder model, i.e. cryptographic primitives, algebraic properties, and intruder capabilities. In particular, our formalization works at the level of messages and is thus independent of the technical and conceptual details of the overall protocol model (which could be formalized, for example, by multiset rewriting, strand spaces, or process calculi). Moreover, our approach allows us to consider multiple guesses simultaneously.

Our formalization is based on the idea of explicitly representing the intermediate states of the computation of off-line guessing attacks. We call these intermediate computation states *maps*. Intuitively, a map is the result of the intruder constructing a message from his knowledge and inserting, in place of an unknown value, every value from the dictionary he uses to perform the attack; it thus maps candidate values for guessable data to the concrete message that would result in the respective cases.

Organization. In Section 2 we discuss the intuitions underlying our approach to off-line guessing, and in Section 3 we give a deduction system that captures these intuitions. As a concrete example, in Section 4 we consider Microsoft's Challenge/Response Authentication Protocol, version 2 (MS-CHAPv2). We draw conclusions and discuss related and future work in Section 5.

2 Context and Intuitions

The Dolev-Yao intruder model has proved to be an appropriate abstraction in the formal analysis of security protocols, as it idealizes the behavior and properties of cryptographic operations. However, when considering off-line guessing, this model suffers from several problems — even though guessing has nothing to do with "breaking" cryptography. In this section, we illustrate the intuitions behind our formalization of off-line guessing. To that end, we first describe the shortcomings of the Dolev-Yao model, in particular with respect to guessing, and then argue how it can be appropriately modified to faithfully model a "semi-ideal world" in which cryptography is still perfect but passwords are guessable. This will lead us to introducing the notion of maps, which explicitly represent the intermediate computation steps of an intruder during off-line guessing.

2.1 Problems with the Standard Dolev-Yao Intruder Model

Most approaches to the formal analysis of security protocols abstract away from the details of real cryptography and instead use a term-algebra of messages in the style of the Dolev-Yao intruder model [14], where concrete data like agent names, nonces, or keys are represented by constants, and cryptographic operations like different kinds of encryption are represented by function symbols. Inherent in the models considered in such approaches are several (closely related) simplifying assumptions:

- Cryptography is perfect, i.e. there is no way to decrypt messages without knowing the proper key.
- Syntactically different terms represent different messages. This assumption is often relaxed by defining algebraic properties, e.g. $(a^b)^c = (a^c)^b$, which induce an equivalence relation on the terms, and one then considers the quotient algebra in which syntactically different terms represent the same value if and only if they are equivalent.

- It is not defined what happens when someone attempts to decrypt an encrypted message using a wrong key, i.e. by explicitly modeling that in this case one obtains a string of random bits.
- The intruder always implicitly knows the format or structure of messages being exchanged in a protocol, even if he cannot decrypt them.

The Dolev-Yao intruder model, which incorporates these simplifying assumptions, has proved to be extremely successful in security protocol analysis. However, these assumptions also have many subtle implications. Firstly, when an agent creates a "fresh" nonce, this is usually modeled by picking a fresh constant symbol, which is then "by definition" different from anything that was seen before. One thus abstracts away the small chance that an agent accidentally "generates" a nonce that was used before (maybe by a different agent). Secondly, the cryptographic operations behave like injective functions, for instance $\{\!|m|\!\}_k \neq \{\!|m'|\!\}_{k'}$ if $k \neq k'$ or $m \neq m'$. This abstracts away the small probability that different operations may lead to the same result, e.g. that encrypting different plain-texts with the same key may produce the same bit-string. This leads, for instance, to the unrealistic consequence that all hash-functions are perfect, e.g. never produce the same hash-value for different messages.

These assumptions have unrealistic and counter-intuitive consequences, but still they often provide an appropriate abstraction level for analysis: "collisions" like the ones described above usually cannot be systematically exploited by an intruder, at least not if we assume a reasonable behavior of the cryptographic operations. For example, it should be computationally difficult (though, of course, not impossible) to produce different messages with the same hash-value. The Dolev-Yao model can thus indeed be effectively used to abstract away the possibility that the intruder exploits properties of real cryptography. However, when we consider guessing attacks, we can observe further clashes with the assumptions of the idealized Dolev-Yao model: the intruder may observe a message that is encrypted with a password, and if this password is poorly chosen, it may appear in a dictionary that the intruder possesses; so, roughly speaking, he "knows" the password. However, if we assume that the password is simply part of the intruder knowledge, then, according to the Dolev-Yao model, the intruder can decrypt everything encrypted with that password. This is, however, not the case in reality: although the intruder possesses a dictionary that contains the password, he does not know which entry in the dictionary it is, and it might be impossible to identify the right entry. In other words, not distinguishing between guessable and known messages in the Dolev-Yao model is like assuming that the intruder "always guesses correctly".

2.2 Rule-based Extensions of the Dolev-Yao Intruder Model

Despite all these difficulties, it is still attractive to use a model in the style of Dolev and Yao in order to abstract from the details of real cryptography, not only for "standard" protocol analysis but also when considering a guessing intruder. In this way, several approaches, e.g. [10, 12, 13, 17], propose extensions

of a Dolev-Yao-style model for off-line guessing, where the guessable messages (i.e. those that occur in the intruder's dictionary) are not part of the initial intruder knowledge, but off-line guessing is modeled by additional rules that describe the intruder's ability to make a guess and verify it. An example of such a rule is:

- If the intruder knows a message m and its symmetric encryption $\{\!|m|\!\}_{pw}$ with a password pw, and pw is guessable,
- then he can verify his guess and thus obtain pw (and add it to his knowledge).

It is easy to convince oneself that such a rule is correct with respect to one's intuition: the intruder can generate, for every pw' in his dictionary, the term $\{\!|m|\!\}_{pw'}$ and compare each result with the original message $\{\!|m|\!\}_{pw}$. Assuming that there are no collisions, there is only one entry in his dictionary, namely pw, such that the constructed message is identical with the original one.

However, it is unclear how we can ever be sure that a given set of such rules is complete, i.e. that all the ways to find out messages by guessing are covered by the rule set. To see how hard it is to find a complete set of rules, consider the following rule, which captures another aspect of guessing:

- If the intruder knows the symmetric encryption $\{\!|\langle m_1, m_2\rangle|\!\}_{pw}$ with a guessable password pw of a composed message $\langle m_1, m_2\rangle$ and he knows the first component m_1,
- then he can obtain pw (and thus also m_2).

This is the case because the intruder can attempt to decrypt the given message with every entry in his dictionary as the decryption key, and — again abstracting from collisions — only for one entry of the dictionary, the decrypted message begins with m_1, which he can check as he knows m_1. This identifies the right entry pw in his dictionary and he obtains the second component m_2.

2.3 An Explicit Model of Off-line Guessing

Obviously, one can find many such rules, and even when one arrives at a set of rules that seem to cover one's intuition completely, it is unclear how to prove their completeness, as there is no formal definition of the underlying concept that should be captured by these rules. Also, when extending the model with new cryptographic operators, e.g. exponentiation, one would have to appropriately extend the set of rules for off-line guessing as well.

However, we propose that there is a simple intuition behind the two rules above and similar ones, which is based on operations the intruder can perform on every entry in his dictionary so that he can uniquely determine one particular entry of the dictionary (which then represents the "correct" guess). Our idea to formalize off-line guessing attacks is to formalize this intuition by making explicit the intermediate states of such a computation, which we express via *maps*. Roughly speaking, a map relates each entry of the dictionary (or n-tuples thereof) to the outcome of a sequence of operations under this guess (or these guesses).

v	$\{\!\lvert m \rvert\!\}_v$
d_1	$\{\!\lvert m \rvert\!\}_{d_1}$
d_2	$\{\!\lvert m \rvert\!\}_{d_2}$
$\vdots$	$\vdots$
d_n	$\{\!\lvert m \rvert\!\}_{d_n}$

v_1	v_2	$\{\!\lvert v_2 \rvert\!\}_{v_1}$
d_1	d_1	$\{\!\lvert d_1 \rvert\!\}_{d_1}$
d_1	d_2	$\{\!\lvert d_2 \rvert\!\}_{d_1}$
$\vdots$	$\vdots$	$\vdots$
d_n	d_n	$\{\!\lvert d_n \rvert\!\}_{d_n}$

v	$\pi_1(\{\!\lvert\{\!\lvert m_1, m_2 \rvert\!\}_{pw}\rvert\!\}_v)$
d_1	$\pi_1(\{\!\lvert\{\!\lvert m_1, m_2 \rvert\!\}_{pw}\rvert\!\}_{d_1})$
d_2	$\pi_1(\{\!\lvert\{\!\lvert m_1, m_2 \rvert\!\}_{pw}\rvert\!\}_{d_2})$
$\vdots$	$\vdots$
d_n	$\pi_1(\{\!\lvert\{\!\lvert m_1, m_2 \rvert\!\}_{pw}\rvert\!\}_{d_n})$

Fig. 1. Three examples of maps

Fig. 1 shows three examples of maps, where the intruder's dictionary contains the entries $\{d_1, \ldots, d_n\}$. The first is the map for the example $\{\!\lvert m \rvert\!\}_{pw}$ covered above, where pw appears in the dictionary, i.e. $d_c = pw$ for the "correct" entry $c \in \{1, \ldots, n\}$: the intruder encrypts the known message m with each of the d_i. Only one of the entries $\{\!\lvert m \rvert\!\}_{d_i}$ corresponds to the original message $\{\!\lvert m \rvert\!\}_{pw}$, and this entry reveals the correct password $pw = d_c$. The second example demonstrates that the concept of maps can be easily extended to multiple guesses: if the intruder knows a term $\{\!\lvert pw' \rvert\!\}_{pw}$ which consists of a guessable message encrypted with a guessable key-term, then the intruder can simply build every encryption of an entry in his dictionary with an entry in his dictionary.

Before we turn to the third example displayed in Fig. 1, we need to consider in more detail one of the simplifying assumptions mentioned above: what happens when encrypted messages are decrypted with a wrong key.

2.4 Decryption with a Wrong Key

In a Dolev-Yao intruder model without guessing, it is not necessary to consider what happens when decrypting messages with a wrong key; this is because the intruder knows whether he knows a particular key. Under the perfect cryptography assumption, he will thus not even attempt to decrypt messages for which he does not know the proper key. Also, we can assume that the intruder will send to honest agents only messages that they can receive, i.e. messages that are encrypted with the keys that the honest agents will use for decryption, since the intruder will not gain anything from a situation where an agent reads some random bits after decryption of incoming messages. When considering guesses, however, it is important to model what happens if the intruder decrypts messages with a wrong key, for instance because he might simply try out every entry of his dictionary as the decryption key.

It has become standard in Dolev-Yao-style intruder models to have explicit rules for decryption of messages, i.e. of the form

- If the intruder knows $\{\!\lvert m \rvert\!\}_k$ and k,
- then he can obtain m by decryption.

Since these rules require that the intruder knows the key k, they are not really appropriate when considering guessed keys. Rather, we want to turn to the original form of the Dolev-Yao model where the intruder can perform encryption

and decryption operations using as keys any composition of messages he knows, and we add algebraic equations which, for instance, express that symmetric encryption and decryption with the same key cancel each other out, i.e.

$$\{\!|\{\!|m|\!\}_k|\!\}_k = m\ .$$

Note that, for the sake of simplicity, we have assumed here that symmetric encryption and decryption are in fact the same procedure (as is, for instance, the case for exclusive-or); this is not a restriction but may introduce unrealistic attacks in general.

The reason why we turn to this more complicated model with cancellation equations is that the term $\{\!|\{\!|m|\!\}_k|\!\}_{k'}$ is equal to the term m if and only if $k = k'$ (assuming that there are no other algebraic properties for the operation $\{\!|\cdot|\!\}_\cdot$). This is exactly what we want, as the intruder can now attempt the decryption with several keys, which results in the original message m only in the case that he used the right key. In all other cases, he obtains a term of the form $\{\!|\{\!|m|\!\}_k|\!\}_{k'}$, which represents some random bits he obtained after decryption, but which he is not a priori able to distinguish from the "real" message m.[1]

We can now return to the third example of Fig. 1, where the intruder tries to find out the guessable password pw of the message $\{\!|\langle m_1, m_2\rangle|\!\}_{pw}$ where he knows the message m_1. Here, we denote with π_1 and π_2 the projections to the first and second component of a pair, i.e. we have the cancellation rules $\pi_1(\langle m_1, m_2\rangle) = m_1$ and $\pi_2(\langle m_1, m_2\rangle) = m_2$. The intruder builds a map in two steps: first he encrypts the given message $\{\!|\langle m_1, m_2\rangle|\!\}_{pw}$ with every entry of the dictionary, and then he builds the projection to the first component. Only for the correct guess, i.e. the entry c with $d_c = pw$, the resulting message has the property

$$\pi_1(\{\!|\{\!|\langle m_1, m_2\rangle|\!\}_{pw}|\!\}_{d_c}) = \pi_1(\langle m_1, m_2\rangle) = m_1\ ,$$

and thus there is exactly one entry in this map that equals m_1, while all other entries are "irreducible" terms in the sense that they are not equal to any simpler term. Note that the intruder could not verify the correct entry the way we showed, if he did not initially know the message m_1 (and there is no other way to verify the guess).

To summarize, we have so far described a model where the intruder can compose maps by applying operations on every term in a dictionary and on messages that he knows for sure, and compare the outcome of each entry in the map with other messages he knows. Only in the case that there is a single entry in the map that he can distinguish from the other ones is the guessing successful in the sense that he can identify the correct value(s) of the guess(es). Observe that this model is independent from the concrete set of operators and algebraic equations considered (we have only used them for concrete examples) and we can

[1] Note that in the case that symmetric encryption and decryption are the same algorithm as in our example, the intruder might even construct the term $\{\!|\{\!|\{\!|m|\!\}_k|\!\}_{k'}|\!\}_{k'}$ by re-encrypting the term with the same key he used for decryption and, modulo the cancellation, obtain the term he started with, i.e. $\{\!|m|\!\}_k$.

thus entirely avoid specifying a large set of rules to capture all circumstances in which an intruder can guess certain messages. In fact, we have only two "rules": firstly, the intruder can arbitrarily compose messages and maps that he already knows to form new maps, and secondly, he can check if there is a particular entry in a map. In fact, as we will explain in the next section, we do not need to distinguish between messages and maps any more, as we can regard any message as a special case of a map with only one entry.

Before we introduce our formalization of off-line guessing, let us conclude this section with some remarks. First, observe that the above discussion does not contain any notion of the probability that a guessing attack is successful. Furthermore, it does not depend on the size of the dictionary, and ignores the case that a password might be guessable (in the sense that it stems from a small set of values), but is not contained in the concrete dictionary of the intruder. The reason why these technical details can be ignored is that one might consider a protocol as vulnerable if there is the mere possibility to mount an attack. In fact, although the vulnerability of a protocol to off-line guessing attacks can be modeled by expressing explicitly the probability of the intruder guessing correctly, in our model we are only concerned with expressing whether this vulnerability exists or not. Hence, a protocol that is vulnerable to off-line guessing according to our model might still be "safe" in practice if the users choose good passwords that cannot be easily guessed so that the attack becomes infeasible. Finally, one may wonder what consequences would arise if we consider an intruder with several dictionaries (instead of just one). It is, however, not difficult to see that this cannot affect our notion of a guessing attack as one could consider the union of all such dictionaries as *the* intruder's dictionary.

3 A Formal Model for Off-Line Guessing

We now introduce our formalization of off-line guessing: we give a Dolev-Yao-style deduction system that attempts to capture the intuitions behind off-line guessing in a uniform and general way. As we remarked above, our formalization works at the level of messages and is thus independent of the overall protocol model and of the details of the considered intruder model, i.e. cryptographic primitives, algebraic properties, and intruder capabilities. Moreover, it allows us to consider multiple guesses simultaneously.

Definition 1. *An* intruder model $\mathcal{I} = (\Sigma, \mathcal{O}, \approx, \mathcal{D})$ *consists of a finite set* Σ *of operators where* $\Sigma_0 \subseteq \Sigma$ *is a set of constants (i.e. symbols of arity* 0*), a subset* $\mathcal{O} \subseteq \Sigma$ *of these operators, called the* intruder-accessible *operators, a congruence relation* $\approx \subseteq \mathcal{T}_\Sigma^2$ *on terms (usually defined through a set of equations), and a finite dictionary* $\mathcal{D} \subseteq \Sigma_0$*, where* $\mathcal{D} \cap \mathcal{O} = \emptyset$ *and* $d_1 \not\approx d_2$ *for any pair of constants* $d_1, d_2 \in \mathcal{D}$ *with* $d_1 \neq d_2$.[2] *Also, let* V *be a set of variables disjoint from symbols in* Σ.

[2] For convenient modeling of fresh data, the signature Σ may contain infinitely many constants, as long as the dictionary $\mathcal{D}$ is finite, without affecting the results presented here. We assume, however, a finite signature for simplicity.

We assume that the reader is familiar with standard concepts like terms and substitutions (see, for instance, [3]). Given a substitution σ and a term t, we denote the application of σ to t by writing $t\sigma$, the domain of σ by writing $\mathrm{dom}(\sigma)$, and the variables of t by writing $\mathrm{vars}(t)$. In this paper, we consider only substitutions of variables with guesses (i.e. elements of the dictionary $\mathcal{D}$).

Example 1. As a running example, we will consider the intruder model

$$\mathcal{I}_e = (\Sigma_e, \mathcal{O}_e, \approx_e, \mathcal{D}_e)\,,$$

where the signature Σ_e consists of a set of constants Σ_{0e} and the following operators: asymmetric encryption $\{\cdot\}_\cdot$, the inverse key of an asymmetric key-pair $\cdot^{-1}$, symmetric encryption $\{\!|\cdot|\!\}_\cdot$, function application $\cdot(\cdot)$, and pairing $\langle\cdot,\cdot\rangle$, as well as the two corresponding projections $\pi_1(\cdot)$ and $\pi_2(\cdot)$. All operators besides Σ_{0e} and $\cdot^{-1}$ are accessible to the intruder, i.e. $\mathcal{O}_e = \Sigma_e \setminus (\Sigma_{0e} \cup \{\cdot^{-1}\})$. Equivalence on terms is equality in the quotient algebra $\mathcal{T}_{\Sigma_e}/{\approx_e}$ for the following equations:

$$\begin{gathered}\{\{m\}_k\}_{k^{-1}} \approx_e m\,, \qquad \{\!|\{\!|m|\!\}_k|\!\}_k \approx_e m\,, \qquad (k^{-1})^{-1} \approx_e k\,,\\ \pi_1(\langle m_1, m_2\rangle) \approx_e m_1\,, \qquad \pi_2(\langle m_1, m_2\rangle) \approx_e m_2\,.\end{gathered}$$

The dictionary $\mathcal{D}_e$ consists of a set of constants $\mathcal{D}_e = \{d_1, \ldots, d_n\}$. □

Our formalization is based on the notion of *maps* introduced in the previous section, as illustrated in Fig. 1. As we remarked above, a map is the result of the intruder constructing a message from his knowledge and inserting, in place of an unknown value, every value from the dictionary he uses to perform the attack. The main characteristic of a map is that it is uniform and complete, in the sense that exactly the same sequence of operations is performed for the various values of the guesses, and that all possible guesses in the dictionary are considered. In order to formally define what we mean by "the same sequence of operations" of a map, we introduce the notion of a *pattern term*:

Definition 2. *Let $\mathcal{I} = (\Sigma, \mathcal{O}, \approx, \mathcal{D})$ be an intruder model. A* pattern term *(or, simply,* pattern*) P is simply an element of $\mathcal{T}_\Sigma(V)$. We say that a set M is a map iff there is a pattern term P such that*

$$M = \{(\sigma, P\sigma) \mid \mathrm{dom}(\sigma) = \mathrm{vars}(P) \wedge \forall v \in \mathrm{vars}(P).\ v\sigma \in \mathcal{D}\}\,.$$

Note that this formal definition represents exactly the way we described maps informally in the previous section, namely as tables of guesses and the corresponding outcomes. For instance, each of the three maps displayed in Fig. 1 is represented by the term on top of the right-most column, i.e. $\{\!|m|\!\}_v$, $\{\!|v_2|\!\}_{v_1}$, and $\pi_1(\{\!|\{\!|\langle m_1, m_2\rangle|\!\}_{pw}|\!\}_v)$, respectively.

Note also that it follows straightforwardly from the definition that there is a bijection between maps and patterns, modulo equivalence of patterns and renaming of variables. Hence, a map is uniquely represented by a pattern that is parameterized over a set of data to be guessed, and the intruder's computation

of the outcome of this pattern for every possible combination of candidate values for the guesses.

Given this bijection, in the following we will identify maps and their corresponding patterns. In particular, for a map M and the corresponding pattern P, we will identify entries of M and substitutions of variables with guesses in P, as well as look-up in M and the term resulting from the application of a substitution of variables with guesses in P.

Maps are expressive constructs: we can view all messages as maps with just one entry, represented by a ground term (so the only substitution possible is the identity). Also, the dictionary itself is a map, which is represented by a variable v. This expressiveness allows us to consider a formal model where the intruder knowledge consists only of maps.

We now define an entailment relation over maps, which tells us when the intruder can derive a map from a set of maps. In particular, given that we identify maps with corresponding patterns, we define the entailment over patterns for simplicity. This entailment relation yields a deduction system. Given that "normal" messages are a special case of maps, we can see this system as an extension of (a system for) the standard Dolev-Yao intruder. We therefore call this system the *guessing Dolev-Yao intruder* and denote the relation by $\cdot \in \mathcal{GDY}_{\mathcal{I}}(\cdot)$.

Definition 3. *Let $\mathcal{I} = (\Sigma, \mathcal{O}, \approx, \mathcal{D})$ be an intruder model. The* entailment relation *$P \in \mathcal{GDY}_{\mathcal{I}}(\mathcal{P})$, which expresses that the set of patterns $\mathcal{P}$ entails the pattern P, is the smallest relation closed under the following rules:*

$$\frac{}{P \in \mathcal{GDY}_{\mathcal{I}}(\mathcal{P})}\ \text{AX}P\ (\text{for } P \in \mathcal{P}), \qquad \frac{}{v \in \mathcal{GDY}_{\mathcal{I}}(\mathcal{P})}\ \text{AX}v\ (\text{for } v \in V),$$

$$\frac{P_1 \in \mathcal{GDY}_{\mathcal{I}}(\mathcal{P}) \quad \cdots \quad P_n \in \mathcal{GDY}_{\mathcal{I}}(\mathcal{P})}{op(P_1, \ldots, P_n) \in \mathcal{GDY}_{\mathcal{I}}(\mathcal{P})}\ \text{OP}\ (\text{for } op \in \mathcal{O} \text{ with arity } n),$$

$$\frac{P_1 \in \mathcal{GDY}_{\mathcal{I}}(\mathcal{P}) \quad P_2 \in \mathcal{GDY}_{\mathcal{I}}(\mathcal{P})}{v\sigma \in \mathcal{GDY}_{\mathcal{I}}(\mathcal{P})}\ \text{VER},$$

where the rule VER *has the following side conditions:*

- $v \in \text{vars}(P_1) \cup \text{vars}(P_2)$,
- $\Theta = \{\sigma \mid \text{dom}(\sigma) = \text{vars}(P_1) \cup \text{vars}(P_2) \ \wedge\ \forall v \in \text{dom}(\sigma).\ v\sigma \in \mathcal{D}\}$,
- $\sigma \in \Theta$,
- $P_1\sigma \approx P_2\sigma$,
- $\forall \sigma' \in \Theta.\ \text{different}_{P_1,P_2}(\sigma, \sigma') \implies P_1\sigma' \not\approx P_2\sigma'$, *where* $\text{different}_{P_1,P_2}(\sigma, \sigma')$ *stands for:* $P_1\sigma \not\approx P_1\sigma' \vee P_2\sigma \not\approx P_2\sigma'$,
- $\forall P' \in \mathcal{T}_{\Sigma}(V).\ P' \approx P_1 \implies \text{vars}(P') \supseteq \text{vars}(P_1)$ *and* $\forall P' \in \mathcal{T}_{\Sigma}(V).\ P' \approx P_2 \implies \text{vars}(P') \supseteq \text{vars}(P_2)$.

The first two axiomatic rules are straightforward: a set of patterns $\mathcal{P}$ entails all patterns that are contained within it, and for any guessable value, a set of patterns $\mathcal{P}$ entails a pattern representing an unverified guess of that value

that has the whole of dictionary $\mathcal{D}$ as candidates. (Recall that we consider only substitutions of variables with guesses in $\mathcal{D}$.) The rule OP expresses how patterns can be combined. Given n patterns entailed by $\mathcal{P}$, the rule states that they can be combined using an n-ary operator $op \in \mathcal{O}$ yielding a pattern that is the application of op on the constituent patterns.

As explained in the previous section, the standard Dolev-Yao intruder system includes both synthesis rules for composing messages and analysis rules for decomposing them; see, for example, [4–6]. Examples of the latter type are decryption and the projection of a paired message into its two components. Using the equational theory, however, one can express analysis steps of the intruder as composition using decryption operators (and cancellation of encryption and decryption with corresponding keys).

We now turn our attention to the verification of guesses, rule VER. In general, verifying a guess requires that the intruder can construct two maps that correspond on exactly one entry (we discuss below how we formalize different substitutions, representing different entries, modulo the algebraic properties). Thus, there is one uniquely determined substitution of dictionary entries for the variables in the two corresponding patterns which yields the same message. This substitution corresponds to the correct guesses, and after verifying his guesses, the intruder learns these correct values for the guesses in the classical sense. Note that this simple definition avoids any notion of "different ways to construct the same message" as is often required in other formal approaches to guessing. Rather, the definition implicitly prevents that the intruder can "cheat himself": suppose he tries to verify something by taking two copies of the same map (which is not forbidden); then either the maps do not contain any variables (so there is nothing guessable to obtain) or the two maps correspond on all substitutions.

To illustrate this further, let us consider the case where the intruder has derived two patterns P_1 and P_2, and Θ is the set of all substitutions for the variables of P_1 and P_2 with entries of the dictionary. If there is one substitution $\sigma \in \Theta$ on which the two patterns yield the same message, $P_1\sigma \approx P_2\sigma$, and the patterns differ for all different substitutions, then the intruder has indeed verified the correct value for any of the data that he has guessed in the two patterns/maps. Thus, for any variable $v \in \mathrm{vars}(P_1) \cup \mathrm{vars}(P_2)$ in the two patterns, he can derive the correct value $v\sigma$.

Recall that we identify entries of a map and substitutions of variables with guesses in the corresponding pattern. The notion of different substitutions σ and σ' (in the sense that they represent different entries) is not simply $\sigma \neq \sigma'$ for the following reason. A map may contain several different entries that yield the same value. Consider, for instance, an intruder model that extends our running example $\mathcal{I}_e$ with an operator $\oplus$ representing exclusive-or, which has, among other properties, the property of being commutative. Assume further that the intruder knows the message $P_1 = pw_1 \oplus pw_2$ for two guessable but unknown passwords pw_1 and pw_2. To find out the passwords, the intruder constructs the map $P_2 = v_1 \oplus v_2$. The problem now is that for the two different entries

$\sigma = [v_1 \mapsto pw_1, v_2 \mapsto pw_2]$ and $\sigma' = [v_1 \mapsto pw_2, v_2 \mapsto pw_1]$ we have $P_2\sigma \approx P_2\sigma'$ due to the algebraic properties, and thus there are several entries on which the two maps P_1 and P_2 correspond. Therefore, our notion of different entries is relative to two maps that we want to compare, and we thus consider those pairs of substitutions σ and σ' as being different entries relative to the maps P_1 and P_2 if they indeed make a difference in at least one of the maps, i.e. if $P_1\sigma \not\approx P_1\sigma'$ or if $P_2\sigma \not\approx P_2\sigma'$.

Finally, note that the last pair of side conditions of the rule VER ensures that for the verification the intruder uses only maps P_1 and P_2 that are not equivalent to some map P' that comprises fewer variables. For instance, for our example intruder model, the rule VER cannot be applied to the map $\{\!|\{\!|m|\!\}_v|\!\}_v$ as this map is equivalent by $\approx_e$ to the map m. In other words, this condition expresses that we don't consider maps with redundant variables, i.e. those on which the outcome does not depend.

The formal model presented here only allows for derivations that are indeed possible according to the intuition given in the previous section. On the other hand, all operations the intruder could perform according to this intuition have a counter-part in the formal model, since composition and decomposition of messages with guesses is described by respective operations on maps/patterns, and verification of guesses is described by the comparison of maps/patterns. But, of course, one can only informally justify that a formal model captures the intuition underlying it.

Example 2. We illustrate with the examples from the previous section, using our paradigmatic intruder model $\mathcal{I}_e = (\Sigma_e, \mathcal{O}_e, \approx_e, \mathcal{D}_e)$. In the first example, the intruder knowledge consists of a set of patterns $\mathcal{P}$ that contains the messages $\{\!|m|\!\}_{pw}$ and m, and $pw \in \mathcal{D}_e$ is guessable. The intruder first constructs the map $\{\!|m|\!\}_v$ by encrypting m with every entry of the dictionary, and then compares it with the original message $\{\!|m|\!\}_{pw}$ to verify his guess of pw. This comparison is possible since there is exactly one entry in the dictionary that equals pw and no other substitution for v that can make $\{\!|m|\!\}_{pw}$ and $\{\!|m|\!\}_v$ equal:

$$\dfrac{\dfrac{}{\{\!|m|\!\}_{pw} \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}P \quad \dfrac{\dfrac{}{v \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}v \quad \dfrac{}{m \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}P}{\{\!|m|\!\}_v \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{OP}}{pw \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{VER}$$

In the second example, the intruder knowledge $\mathcal{P}$ contains $\{\!|m_2|\!\}_{m_1}$, but neither m_1 nor m_2. Further, both m_1 and m_2 are guessable, i.e. $m_1, m_2 \in \mathcal{D}_e$. The intruder first encrypts every entry of the dictionary with every entry of the dictionary to obtain the pattern $\{\!|v_2|\!\}_{v_1}$, which he can then compare to $\{\!|m_2|\!\}_{m_1}$. Since again only the substitution with the correct guesses can make the two terms equal, he obtains both m_1 and m_2 by this comparison. Simplifying the presentation, we have joined the derivation of the two messages into one tree:

1. $A \to S : A$
2. $S \to A : Ns$
3. $A \to S : Na, H(Pw, Na, Ns, A)$
4. $S \to A : H(Pw, Na)$

Fig. 2. The MS-CHAPv2 Protocol

$$\dfrac{\dfrac{}{\{\!|m_2|\!\}_{m_1} \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}P \qquad \dfrac{\dfrac{}{v_1 \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}v \quad \dfrac{}{v_2 \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}v}{\{\!|v_2|\!\}_{v_1} \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{OP}}{m_1, m_2 \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{VER}$$

In the third example, the intruder knowledge $\mathcal{P}$ contains the messages m_1 and $\{\!|\langle m_1, m_2\rangle|\!\}_{pw}$, and pw is guessable. The intruder tries to decrypt this message with every entry in his dictionary, obtaining the pattern $\{\!|\{\!|\langle m_1, m_2\rangle|\!\}_{pw}|\!\}_v$, and then builds the projection to the first component for every entry, which yields the pattern $\pi_1(\{\!|\{\!|\langle m_1, m_2\rangle|\!\}_{pw}|\!\}_v)$. He compares this pattern with the known message m_1 and only for the right guess $v = pw$ the pattern yields this value, so he obtains pw:

$$\dfrac{\dfrac{}{m_1 \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}P \qquad \dfrac{\dfrac{\dfrac{}{v \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}v \quad \dfrac{}{\{\!|\langle m_1, m_2\rangle|\!\}_{pw} \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}P}{\{\!|\{\!|\langle m_1, m_2\rangle|\!\}_{pw}|\!\}_v \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{OP}}{\pi_1(\{\!|\{\!|\langle m_1, m_2\rangle|\!\}_{pw}|\!\}_v) \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{OP}}{pw \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{VER}$$

□

4 A Concrete Example: The MS-CHAPv2 Protocol

As a concrete example, we consider an entire protocol, namely Microsoft's Challenge/Response Authentication Protocol, version 2 (MS-CHAPv2 [21]). MS-CHAPv2 is the authentication mechanism for the Point-to-Point Tunneling Protocol (PPTP [16]), which itself is used to secure PPP connections over TCP/IP. It is well known that this protocol is vulnerable to off-line guessing attacks [20], and we illustrate how one can easily detect this vulnerability using our approach.

Fig. 2 shows an abstracted version of the MS-CHAPv2 Protocol in the Alice& Bob-style notation that is standard in the literature. Note that, for simplicity, we refrain here from explicitly displaying the pairing operator and simply use commas. The protocol should achieve mutual authentication between a client A and server S based on an initially shared password Pw, which we of course assume to be guessable.

$$\dfrac{\dfrac{}{h(\langle pw, na\rangle) \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}P \qquad \overset{\Pi}{h(\langle v, na\rangle) \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}}{pw \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{VER}$$

where Π is

$$\dfrac{\dfrac{}{h \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}P \qquad \dfrac{\dfrac{}{v \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}v \qquad \dfrac{}{na \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{AX}P}{\langle v, na\rangle \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{OP}}{h(\langle v, na\rangle) \in \mathcal{GDY}_{\mathcal{I}_e}(\mathcal{P})}\ \mathrm{OP}$$

Fig. 3. An example derivation using the deduction system of Section 3

As an illustrative case, let us consider the situation in which the intruder has observed a single run of the protocol between two honest agents a (playing in the role A) and s (playing in the role S). Then the intruder has the following knowledge (assuming that he knows the hash-function h):

$$\mathcal{P} = \{a, s, h, na, ns, h(pw, na, ns, a), h(pw, na)\}\ .$$

Note that we have used lower-case letters to distinguish the concrete data of a protocol run from the protocol variables. Fig. 3 shows that, from this knowledge, the intruder can indeed derive pw, if it is guessable.

He simply generates a map for the hash-value of the message under all values of his dictionary in place of pw, and compares the outcome with the observed message. Note that a similar attack would have already been possible after observing only the first three messages of the protocol, since the intruder can similarly build the hash-value of message three under all guesses for the password. Note also that, as we remarked above, in our model we only investigate whether a protocol is vulnerable to off-line guessing or not. Hence, if the users choose good passwords that cannot be easily guessed, it might be infeasible to exploit in practice the vulnerability of the MS-CHAPv2 Protocol that we have just described.

To conclude the section, observe that all examples we have given in this paper are examples of the intruder successfully attacking a protocol by off-line guessing. However, even though we lack space to go into the details, we observe that we can also use our formalization as the basis to prove the absence of vulnerabilities of a protocol even under off-line guessing, both in model-checking approaches (where we usually bound the number of sessions considered) or in theorem-proving ones (where we can inductively show the correctness for all situations).

5 Related Work and Concluding Remarks

We have given a formalization of off-line guessing (in the context of the idealized Dolev-Yao intruder model) which is based on the notion of using maps to explicitly represent the intermediate computation steps of an intruder during guessing. Our formalization is general and uniform in the sense that it is

independent of the overall protocol model and of the details of the considered intruder model, i.e. cryptographic primitives, algebraic properties, and intruder capabilities. Moreover, it allows us to consider multiple guesses simultaneously.

Several other approaches have similarly considered extensions of the Dolev-Yao model with additional intruder capabilities to model a notion of off-line guessing, starting with the work of [17], which inspired [10, 12] and was extended in [18]. These first approaches are somewhat limited (for instance [12, 17] consider only single guesses) and are thus helpful to find protocol attacks, but are not fully appropriate for providing a definition of off-line guessing and verifying that a protocol is invulnerable to such guessing attacks.

The more advanced approach of [13] presents a theory of off-line guessing that overcomes such limitations and also includes an analysis of its complexity and a basis for an efficient implementation using a symbolic intruder model. Like previous approaches, it is, however, specialized to the standard Dolev-Yao intruder under a free term algebra, and it is based on involved syntactic concepts like normalized intruder derivations to ensure that the guessing extension is not "too powerful" (in the sense that the intruder could derive almost everything that is guessable). The approach of [11] is the closest to ours as it also employs cancellation equations rather than explicit decryption rules. Moreover, like our approach, it is not specialized to a particular intruder model, although it is based on an explicit formalization of protocols using the applied pi calculus of [2] where security properties are defined using a notion of indistinguishability of processes. In [1], the authors show that the intruder deduction problem in this formalism is undecidable unless one imposes strong restrictions on the considered equational theory, and we believe that similar properties hold also for our formalization. It is however not easy, at least in our opinion, to estimate whether the approaches of [11, 13], as well as the previous ones, indeed faithfully formalize the notion of off-line guessing, as it is not completely clear what notion they capture. In essence, these works formally define derivation systems but give little intuition about how their respective systems relate to a common-sense understanding of off-line guessing.

We believe that one of the major contributions of our work is that it describes and formalizes such a common-sense intuition of off-line guessing in a precise way and thus provides an appropriate bridge between the intuitions and a formal model of them. A detailed investigation of our intuition led us to the notion of maps on which our formalization is based. In fact, one might say that maps provide a semantical basis for our rules as they make explicit the intermediate computations of an intruder during an off-line guessing attack. We therefore believe that our formalization is not only uniform and more general, but also conceptually simpler and easier to understand than previous approaches. Note, however, that our approach is based on algebraic equations, and is thus more difficult to implement within a protocol analysis tool than the less expressive approaches. In this paper, we have placed the emphasis on the theoretical foundations underlying our model of guessing, but we are currently working on how our

deduction system can be deployed in practice by integrating it into the symbolic *lazy intruder* model that underlies our protocol model-checker OFMC [4–6].

We conclude by highlighting an open problem that our approach does not address (and that is not mentioned in other works on off-line guessing): when the intruder has verified a guess, for instance, of a password pw belonging to an agent A, then the guessed value is added to his "classical" knowledge. He can then use it, for example, to decrypt other messages. While this is exactly what one might expect, a problem arises when another agent, say B, accidentally has the same guessable password (in theory, there might be several such agents). In this case, in verifying his guess of A's password pw, the intruder has — quite inadvertently — also learned B's password. This problem arises if the intruder did not initially know that there was any correlation between the two passwords. This issue can easily be resolved in the case of passwords, as we can simply model the passwords of different agents by different constants. However, it is not clear how to proceed for other guessable data, like protocol key-words. This problem is related with the question of how much the intruder knows about the format of messages — and the relation of several constants in them. We leave a closer investigation of these issues for future work.

Acknowledgments. We thank Penny Anderson and David Basin for many inspiring discussions, and the anonymous referees for their helpful comments.

References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *Proc. ICALP'04*, LNCS 3142. Springer, 2004.
2. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. POPL'01*. ACM Press, 2004.
3. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge U. Pr., 1998.
4. D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. In *Proc. ESORICS'03*, LNCS 2808. Springer, 2003.
5. D. Basin, S. Mödersheim, and L. Viganò. Constraint Differentiation: A New Reduction Technique for Constraint-Based Analysis of Security Protocols. In *Proc. CCS'03*. ACM Press, 2003.
6. D. Basin, S. Mödersheim, and L. Viganò. OFMC: A Symbolic Model-Checker for Security Protocols. *International Journal of Information Security*. 2004.
7. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proc. EUROCRYPT'00*, LNCS 1807. Springer, 2000.
8. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proc. IEEE Symposium on Security and Privacy 1992*. IEEE Computer Society Press, 1992.
9. E. Bresson, O. Chevassut, and D. Pointcheval. Security proofs for an efficient password-based key exchange. In *Proc. CCS'03*. ACM Press, 2003.
10. E. Cohen. Proving cryptographic protocols safe from guessing attacks. In *Proc. Foundations of Computer Security'02*. 2002.
11. R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. In *Proc. WISP'04*. Electronic Notes in Theoretical Computer Science, 2004.

12. R. Corin, S. Malladi, J. Alves-Foss, and S. Etalle. Guess what? Here is a new tool that finds some new guessing attacks (extended abstract). In *Proc. WITS'03*. Dip. Scienze dell'Informazione, Università di Bologna, Italy, 2003.
13. S. Delaune and F. Jacquemard. A theory of guessing attacks and its complexity. Research Report LSV-04-1, Lab. Specification and Verification, ENS de Cachan, France, 2004.
14. D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.
15. L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.
16. K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. RFC 2637: Point-to-Point Tunneling Protocol, July 1999. Status: Informational.
17. G. Lowe. Analysing Protocols Subject to Guessing Attacks. In *Proc. WITS 2002*.
18. G. Lowe. Analysing Protocols Subject to Guessing Attacks. *Journal of Computer Security*, 12(1), 2004.
19. R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594, 1979.
20. B. Schneier, Mudge, and D. Wagner. Cryptanalysis of Microsoft's PPTP authentication extensions (MS-CHAPv2). In *Proc. CQRE'99*, LNCS 1740, Springer, 1999.
21. G. Zorn. RFC 2759: Microsoft PPP CHAP Extensions, Version 2, Jan. 2000. Status: Informational.