From Absence of Certain Vulnerabilities towards Security Proofs — Pushing the Limits of Formal Verification

Michael Backes, Matthias Schunter
IBM Zurich Research Laboratory, Switzerland
{mbc,mts}@zurich.ibm.com

Research Paper

Abstract

The application of formal methods for rigorously validating cryptographic protocols has been getting increasing attention. The de facto standard for modeling such protocols in formal proof systems is the Dolev-Yao model that, e.g., uses abstract encryption instead of cryptographic encryption primitives. The Dolev-Yao model has been originally intended and successfully used for detecting flaws in many protocols. However, recent publications claim to perform actual *proofs* of security using this model, i.e., absence of *any* attack. We doubt this claim and challenge Dolev-Yao-based models as being oversimplified for establishing security proofs against arbitrary attacks.

We substantiate our claim by an example protocol. This protocol has been proven secure in a Dolev-Yao-based model using formal methods. In a later publication, the protocol has been broken by describing a cryptographic attack. The attack was not detected in the formal analysis since any Dolev-Yao-based model only comprises a predefined set of adversary capabilities. The particular attack to break the protocol was not comprised.

The only reliable long-term remedy is to proof resilience against *all* attacks (both known and unknown ones). Recent approaches on cryptographic models of security have already made great progress towards this goal. Unfortunately, proofs in these are more complex and harder to automate. On the short run, it therefore is appropriate to *improve* the quality of formal analysis without striving for complete proofs. This can be achieved by means of evolving a catalog of adversary capabilities. Future formal analysis can then show resilience against any attack in this catalog. We initiate this discussion on an "adversary capability catalog" by providing a cryptographer's wish list. This list that points out several features which approaches based on the Dolev-Yao model or future extensions of it should cover in order to be effective for cryptographic protocol verification.

1 Introduction

Nowadays, cryptographic protocols are getting increasing attention in both theory and practice. In the early days of security research, these protocols were designed using a simple iterative process: someone proposed a protocol, someone else found an attack, the bug was fixed, and so on, until no further attack was found. Today, it is commonly accepted that this "wait-and-fix" approach requires a long time to become effective and hence does not provide much security guarantee. As one of the most prominent examples, we name SSL, in which subtle flaws were discovered even after years of using it, although its security was well-evaluated.

The long-term goal for actual security proofs is to show absence of *any* attack. This includes cryptographic as well as other attacks, based on known as well as unknown adversary strategies.

Unfortunately, such rigorous cryptographic proofs covering the whole mathematical details rapidly become impractical if they go beyond the individual cryptographic primitives. They moreover have to

be done by hand and are hence prone to error. This motivated the use of formal methods for the verification of cryptographic protocols, i.e., protocols should be verified using model checkers or theorem provers. Since current formal proof systems cannot deal with cryptographic details like probabilism, computational restrictions such as polynomially bounded adversaries, and error probabilities, *abstractions* of cryptography are used instead. This yields the well-known notion of *perfect cryptography*. As these abstractions originated from the seminal work of Dolev and Yao [18], this approach is typically referred to as the *Dolev-Yao model*. In Dolev-Yao-based models, the capabilities of an adversary and the properties of the cryptographic primitives are described by an initial set of rules. The attractiveness of this approach for detecting flaws in security protocols is underlined by a large amount of work done by the formal-methods community, cf. the related literature for a comprehensive overview. In recent times however, there is a strong drift to use this model for establishing *proofs* of security, i.e., results in Dolev-Yao-based models are interpreted in the sense that *no* attack exists against the protocol.

We doubt this claim for two reasons. The first reason is that proofs rely on the abstraction that every possible attack can can be derived from the initial set of adversary capabilities, i.e., they assume some kind of completeness. The problems arising here is that certain cryptographic attacks might have been abstracted away, e.g., cryptographic computations that compute faked messages [48, 44] for breaking a protocol. As a consequence, a system where the abstraction is replaced by an actual cryptographic primitive may be susceptible to attacks that are undetectable in the analysis. One way to fix this it to change to models based on cryptographic notions [14, 43, 45] of security, cf. the related literature. Unfortunately, proofs in these models are hard to automate. As a consequence, we will not elaborate on this problem any further.

The second reason why Dolev-Yao-based analysis does not yield proofs is that the Dolev-Yao approach explicitly models the attacker capabilities. I.e., each analysis only considers a given set of attack strategies instead of covering all attacks. We will present one example of a "Dolev-Yao secure" yet vulnerable protocol in detail. The reason why the Dolev-Yao analysis did not detect the vulnerability was that the adversary capability needed for breaking the system was not foreseen and therefore not considered by the analysis. On the long run, this will probably turn out to the use of models capturing the cryptographic notions of security instead.

We believe that Dolev-Yao-based verification is very effective in proving resilience against known adversary strategies. Therefore, we propose a "catalog of adversary capabilities" as a shorter-term remedy. The goal of this paper is to initiate an evolution of a catalog of attack strategies to be used for Dolev-Yao-based verifications. Future enlargements of this catalog can then contribute to higher the security assurance of Dolev-Yao-based analysis.

Loosely speaking, we propose to switch from protocol evolution using trial-and-error to adversary strategy evolution.

1.1 Outline

The paper is structured as follows: In Section 2, we briefly introduce the Dolev-Yao-based models along with related work and the cryptographic justification of the models. Moreover, we discuss several cryptographic approaches to protocol verification, and we point out some recent models that are suited for capturing the cryptographic details but nevertheless allow for formal verification. In the future, these models may become alternatives to the Dolev-Yao-based models. In Section 3 we describe a protocol of Karjoth et. al. [25] that has been proven secure in a Dolev-Yao-based model using formal methods and afterwards has been broken. The reason was that a particular adversary capability was not considered in the Dolev-Yao-based analysis. In Section 4 we initiate the discussion on an "adversary capabilities catalog" by proposing a cryptographer's wish list that points out future research in Dolev-Yao models with the intention to take the approach closer to the cryptographic reality. It mainly consists of additional

capabilities that should be granted to the adversary for reflecting cryptographic possibilities, and of several attacks that should be detected by a formal analysis in Dolev-Yao-based models.

2 Related Literature

2.1 Overview of the Dolev-Yao-based Models

The Dolev-Yao model has been introduced in [18]. It considers cryptographic primitives as operators in a free algebra, and only allows the adversary to apply certain predefined rules within the algebra. For instance, the set of messages considered in the Dolev-Yao model could be given by

$$Messages \quad := \quad Atom \quad | \quad \mathsf{encrypt}(Message, Key)$$

where $Atom$ is some set of so-called atomic messages and $Key \subseteq Atom$ are those atoms used for encrypting and decrypting messages. For a key K, its inverse key is typically denoted by K^{-1}.

In order to make formal verification feasible in models following this approach (Dolev-Yao-based models), the adversary is defined by a set of rules. These rules determine which messages the adversary is allowed to know, i.e., which messages he is allowed to deduce and create based on an observed set of messages B. The set B typically contains those messages that are sent between the participants of a protocol, i.e., the rules define which information the adversary can learn by eavesdropping on the network.

The following rules represent the typical Dolev-Yao attacker for our set of messages. We write $B \vdash M$ to denote that the adversary is allowed to deduce the message M from B. At first, every message that the adversary has eavesdropped can obviously be used, i.e.,

$$M \in B \Rightarrow B \vdash M. \tag{1}$$

At second, if the adversary knows a message M and a key K, then he is allowed to compute the encryption of M under K, i.e.,

$$B \vdash M \wedge B \vdash K \wedge K \in Key \Rightarrow B \vdash \mathsf{encrypt}(M, K). \tag{2}$$

Finally, the adversary can decrypt a ciphertext if he has the corresponding secret key, i.e.,

$$B \vdash \mathsf{encrypt}(M, K) \wedge B \vdash K^{-1} \Rightarrow B \vdash M. \tag{3}$$

Whenever the adversary eavesdrops a new message the set B is extended and the adversary can use the above rules to deduce and create new messages again. The adversary is restricted to sending only those messages that he has already deduced, i.e., he is not allowed to, e.g., guess a message. The proof is then performed by showing that the adversary cannot deduce a secret, e.g., a secret key. Various proof tools can be used for this task, cf. the related literature. This abstraction simplifies proofs of larger protocols considerably. Note that we only described a simple Dolev-Yao-based model here, which only covers encryption and decryption. Typically used Dolev-Yao-based models are more extensive, i.e., they comprise operators for nonce generation, digital signatures, hash functions, or message pairing and splitting. The overall proof technique is not affected by these extensions.

Originally, formal methods used this approach to detect certain attacks in protocols. As one of the most prominent examples, we point out the work of Lowe [31], which used the model checker FDR to discover an attack against the Needham-Schroeder public-key protocol, which was widely believed to be secure at this time. The goal of searching for common attacks is surely a worthy one, as it helps to lift the protocol to a (much) higher level of security. Moreover, an error found in Dolev-Yao-based models always yields an error in the actual cryptographic implementation.

2.2 Prior Work in The Dolev-Yao model and its Cryptographic Justification

Early work using Dolev-Yao-based models for tool-supported proofs was rather specific with respect to the considered issue and formalism, e.g., [35, 32, 26]. More recently, research mainly focused on standard languages, state exploration tools and theorem proving techniques, mainly initiated by the seminal work of Lowe [31], where the general-purpose model checker FDR was used to find a man-in-the-middle attack on the Needham-Schroeder public key protocol [37]. Work since then made progress in applying model checkers [36, 17] as well as theorem provers [40, 19] for the verification of security protocols, and several specialized model checkers have been developed. Besides investigating the actual verification techniques, research also focused on standard languages for expressing security protocols, e.g., the well-known spi-calculus by Adabi and Gordon [1].

Since this whole line of work turned out to be very successful, the interesting question arose whether these abstractions are indeed justified from the view of cryptography, i.e., whether properties proved for the abstractions are still valid for the cryptographic implementation.[1] Abadi et. al. showed in [4, 2] that the Dolev-Yao model is cryptographically faithful at least for symmetric encryption and synchronous protocols. There, however, the adversary is restricted to passive eavesdropping. Another interesting approach has been presented by Guttman et. al. [21], which starts adapting the strand space theory [47] to concrete cryptographic primitives. More precisely, they show that the probability of two executions of the same protocol – either executed in a Dolev-Yao-like framework or using real cryptographic primitives – may deviate from each other at most for a certain bound. However, their results are specific for the Wegman-Carter system so far. Moreover, as this system is information-theoretically secure, its security proof is much easier to handle than asymmetric primitives since no reduction proofs against underlying number-theoretic assumptions have to be made. Some further approaches for special security goals or primitives are [49, 28]. However, there is evidence that the original Dolev-Yao model is not justified in the presence of active attacks, even if provably secure cryptographic primitives are used, cf. [42] for an (admittedly constructed) counterexample.

2.3 Cryptographic Notions of Security

For living up to the probabilistic nature of cryptography, a framework for dealing with actual cryptography necessarily has to be able to deal with probabilistic behaviors, error probabilities and complexity-theoretically bounded adversaries. Based on these requirements, several general definitions of secure protocols were developed over the years, e.g. [20, 34, 8, 29, 43, 24, 13, 45, 14], which are all potential candidates for such a framework. For a comprehensive analysis of security protocols, a suitable model should moreover capture a reactive environment, i.e., continuous interaction with the users and the adversary. Unfortunately, most of the above work does not live up to these requirements in spite of its generality, mainly since it concentrates on the task of secure function evaluation, which does not capture a reactive environment. Currently, the models of Pfitzmann et. al. [43, 45] and Canetti [14], which have been developed concurrently but independently, stand out as the standard models for sound protocol analysis and design.

Their security definition is *simulatability* which captures the notion of a cryptographically secure implementation. Simulatability bridges the gap between abstract specifications and cryptographic implementations, i.e., abstractions which can be shown to simulate a given implementation in a particular sense are known to be sound with respect to the security definitions of cryptography. Currently, such faithful abstractions have already been developed for medium-sized examples comprising secure message transmission, certified mail, or secure key exchange. Moreover, the recently published universally

[1] An initial comparison between Dolev-Yao and cryptographic notions of security can be found in [41].

composable cryptographic library [7] may pave the way to formal verification of large security protocols within these cryptographically faithful models.

3 A Formally Secure yet Vulnerable Protocol

In [25], Karjoth, Asokan, and Gülcü proposed four protocols which aim at protecting the computational results established by free-roaming mobile agents. Roughly, a shopping agent is described that visits several shops and then collects and compares offers for a specific good. One of the main goals to be established is the integrity of offers, i.e., a malicious shop must not be able to modify already existing offers. This property is called *strong forward integrity*. It is important to note that the authors mainly concentrated on motivating and defining the actual protocols and only included brief sketches of the respective security proofs. In the following, we concentrate on the first protocol, called P1 in [25].

In [9], the strong forward integrity property of P1 has been formally verified using the theorem prover Isabelle [38]. The protocol was as usual expressed in the Dolev-Yao-based model, and the modeling and the proof were explained in a detailed way. Very surprisingly in the context of this result, an attack on P1 was found in [46] which violates the strong forward integrity property. Even more surprisingly, this attack was not a "bit-twiddling" attack with only questionable use in practice, but the attack is very easy to accomplish and succeeds with probability one. In the following, we sketch the protocol P1, its modeling in the formal Dolev-Yao framework, the actual attack on the protocol, and we finally analyze why this flaw has not been detected in this model.

3.1 Sketch of the Protocol

We start by introducing the necessary protocol notation. In the following, we consider an originator S_0, which sends its agent Π to n shops $S_1, \ldots, S_n$ for collecting their offers. The *offer* of S_i is denoted as o_i, the *encapsulated offer* is denoted as O_i.

Let $\mathsf{sig}_i(m)$ denote a digital signature created by S_i for a message m and $\mathsf{enc}_0(m)$ a public-key encryption for m with the public key of S_0. Let H be a one-way collision-free hash function; r_i are randomly chosen nonces of S_i.

The protocol P1 is called the *publicly verifiable chained digital signature protocol*, which is defined as follows:

- Encapsulated Offer:
  - $O_i = \mathsf{sig}_i(\mathsf{enc}_0(o_i, r_i), h_i)$ for $0 \leq i \leq n$
- Chaining Relation:
  - $h_0 = H(r_0, S_1)$
  - $h_i = H(O_{i-1}, S_{i+1})$ for $1 \leq i \leq n$
- Protocol:
  - $S_i \Rightarrow S_{i+1}$: Π, $\{O_k \mid 0 \leq k \leq i\}$ for $0 \leq i \leq n$

The protocol is started by the originator S_0 by picking a random value r_0, computing the hash value h_0 and then constructing the "dummy" encapsulated offer O_0.

Thus, when the agent arrives at shop S_i, it contains the set of previously collected encapsulated offers including an encapsulated offer O_{i-1} from which the next hash value h_i can be computed. In

general, an encapsulated offer O_i contains the offer o_i probabilistically encrypted so that only S_0 can retrieve it. Moreover, it contains a hash of the previous offer O_{i-1} concatenated to the identity of the next shop S_{i+1}. The reason of this is to link the previous offer with the current offer, i.e., it should be impossible to modify O_{i-1} without modifying O_i as well. In fact, even shop S_{i-1} cannot modify its own offer later without invalidating the chain consisting of the O_i. The reasons for including the identity of the next shop is to guarantee that no one other shop than S_{i+1} can append the next offer. The whole sequence of encapsulated offers is called a *chaining relation*.

One of the most important goals is that a malicious shop S_i must not be able to modify already existing offers, i.e., O_k for $k < i$ such that this tampering remains unnoticed by the originator when the agent finally returns. This is called strong forward integrity:

Strong Forward Integrity: None of the shops S_i can modify any encapsulated offers O_k for $k < i$ such that the chain $O_0, O_1 \ldots, O_n$ is still "valid", i.e., such that the originator cannot notice this tampering.

3.2 Formal Method Used

We now briefly describe the Dolev-Yao verification of the strong forward integrity property as performed in [9]. The verification follows Paulson's inductive approach [40], which represents a comprehensive Dolev-Yao-like algebra along with suitable operations for the adversary. Roughly, this algebra augments the algebra which we presented in Section 2.1 with certain cryptographic primitives like abstractions of nonces, hash functions, and digital signatures. Moreover, message pairing is considered. The augmented rules for the adversary then allow to create nonces, hashes and signatures. These rules are *creating rules* (like Equation 2). Moreover, there are *analyzing rules* (like Equation 3), e.g., to split a pair or to extract the message from an encryption given the corresponding secret key.

Using this algebra, the protocol, i.e., the chaining relation can easily be expressed. The proof (of strong forward integrity) is then performed as a typical Dolev-Yao proof, i.e., the set of all messages that the adversary can create and analyze is computed and it is shown that none of these messages can be used to mount a successful attack against the strong forward integrity property.

In the following section, we will describe an attack against the protocol, which violates the strong forward integrity property. The attack is very simple and does not rely on cryptanalysis.

3.3 Attack

In the following, we describe the attack from [46] against the protocol. Assume that S_i is a malicious shop. Then S_i simply picks j at random from $\{1, \ldots, i-1\}$ and a new S_{j+1} of its choice. Note that there is no free choice for S_j once j is fixed, only for S_{j+1}. The key idea of the attack is that S_i uses its own mobile agent Π_{S_i} with its own program to collect offers from S_j. These offers are then later plugged into the chain. Formally, S_i collects an offer from the shop S_j as follows:

$$\begin{aligned} S_i \rightarrow S_j &: \Pi_{S_i}, \{O_0, \ldots, O_{j-1}\} \\ S_j \rightarrow S_{j+1} &: \Pi_{S_i}, \{O_0, \ldots, O_j\} \\ S_{j+1} \rightarrow S_i &: \Pi_{S_i}, \{O_0, \ldots, O_{j+1}\} \end{aligned}$$

When the agent returns to S_i, it throws away O_{j+1}, increments j, and picks a new S_{j+1}. Note that S_i can also repeatedly use different agents until a suitable offer is received. Note further that the "anchor" O_0 of the chaining relation is signed with the secret key of S_0. Hence the chaining relation and encapsulated

offers are build as if S_0's agent had requested the offer, but in reality they have been requested by the malicious shop S_i using its malicious agent Π_{S_i}. If S_i is satisfied with the offers it has collected, it pastes them into S_0's agents and sends it to S_{j+1}. In a nutshell, shops are abused as oracles for generating offers to the terms of the malicious S_i rather than the originator of the protocol S_0.

3.4 Analysis

Why did the formal analysis fail to identify the attack? In [9], the achieved result was perspicuously interpreted in the way that modifying or inserting an offer while preserving the chaining relation requires modifying or inserting as well all the following offers, which is made "*a priori* impossible by asking the shops to sign their offers with their private keys". The answer to the above question also shows why this interpretation is flawed: The protocol does not pay attention to a well-known robustness principle for secure protocol design: "Don't let yourself being used as an oracle for signing or decrypting messages" [5]. In our case shops that have already given their offer can later be used as signing oracles for signing messages by executing another protocol in parallel using another agent. In a nutshell, another agent could be sent to the same shops, these shops will propose new signed offers to this new agent, and these new offers can then safely replace the corresponding offers in the original chaining relation without destroying it. In other words, the attack uses a capability which was not modeled in this particular Dolev-Yao model. Note that this should not be held against the proof in [9], since the author did a good job in explaining and performing his work, but against the underlying model itself, which does not comprise the full range of different attacks. We finally note that this attack can surely be found in more suitable Dolev-Yao model if one adds an additional suitable rule for the adversary. However, besides being more trustworthy than hand-made proofs, this comes close to the wait-and-fix approach again.

3.5 Conclusion

We have illustrated why proofs based on Dolev-Yao-like models should be treated with care. They do not imply provable security in the sense of absence of any attacks, not even absence of simple non-cryptographic attacks. A Dolev-Yao security analysis hence only models a certain set of attacks, which in our case did not comprise the particular attack to which the protocol was vulnerable. We can draw two main conclusions from this: The first is that Dolev-Yao-based verifications do not yield proofs against every attacks. The second is that one should start extending the expressiveness of Dolev-Yao-based models to come closer to the "any attack" that is desirable from a cryptographic perspective.

4 A Cryptographer's Wish list

In contrast to yielding proofs of security, we believe that such models are perfectly suited for detecting flaws, i.e., to increase confidence that a protocol is secure. Moreover, experience shows that proofs done within the model also provide a significant insight in a protocol's possible weaknesses. This also forces the designers to really specify all details and a precise model of the requirements that shall be satisfied. In order to increase the effectiveness of this process, we present a wish list of adversary capabilities that we believe would be desirable to capture in future Dolev-Yao-based extensions. The wish list is mainly motivated by the cryptographic point of view. One should be able to design flawed protocols that exploit exactly one of the vulnerabilities. Together with correct protocols, these protocols can then be used to benchmark the detection rate of a particular formal methodology.

4.1 Realistic Protocol Models

The first step towards better evaluations is to provide realistic modeling of a protocol in the formalism used.

Open-ended Protocols So far, previous work has mostly concentrated on protocols with closed-ended data-structures, where messages exchanged between principals have fixed and finite format. However, in many protocols, the data-structures are *open-ended*, i.e., messages may consist of an a-priori unbounded number of data fields that must be processed in one action. The question how to formally deal with such protocols has been proposed by Meadows [33]. Some results have already been achieved in this topic: The recursive authentication protocol [12] has been subject to formal verification by both Paulson [39], using the theorem prover Isabelle, and by Bryans and Schneider [11], using the PVS theorem prover. Meadows analyzed the protocol in [6] using the NRL analyzer. However, a comprehensive treatment is not yet well understood. Recently, there are also results on decidability issues of the Dolev-Yao-like verification of open-ended protocols [27].

Modeling of Advanced Protocol Assumptions Recently so-called pro-active protocols [15, 23] have been developed which assume that some players may be corrupted at some point in time and then re-initialize themselves and join the honest crowd afterwards. Usually, the desired security requirement then relies on the fact that at most a certain number of players are corrupted at any point in time. To the best of our knowledge, this topic has not been addressed in a formal analysis based on the Dolev-Yao model yet.

4.2 Controlling the Players

Once a protocol has been modeled, an analysis method defines the capabilities of the adversary. In principle, the stronger the adversary, the more reliable the analysis. The weakness that we have detected in Section 3 was caused by the adversary model not reflecting the fact that real-life adversaries can send their own agents.

Determine Whom You Trust for Each Requirement Cryptographic protocols are usually specified by a list of requirements. For each requirement, the authors either describe or assume a set of correct players. The set of correct players can vary by requirement. E.g., for contract signing protocols it is common to assume that the contract verifier is unconditionally correct (otherwise the verification is useless anyway) while the protocols aim at safeguards against cheating notaries even though they are generally assumed to be correct. The different trust assumptions have to be reflected in a Dolev-Yao model.

For evaluating protocols with a potentially unbounded number of participants, it is important that the adversary can determine the number and IDs of participants. Common evaluations assume that, e.g., a two-party protocol involves two parties and an adversary. This may prevent detection of attacks if the adversary is unable to simulate additional protocol players by, e.g., generating certified key sets and then using them in simulated protocol runs.

Adversary drives Participants The adversary should be allowed to largely control the correct participants. This includes starting (sub-)protocols in any order (generating keys, main protocol, recovery), generating keys, inputs, and state-transitions behavior.

The adversary should be allowed to determine all inputs of the protocol. This includes re-using earlier or interleaved protocol messages as an input to a subsequent protocol run. If a protocol, e.g.,

sends a signed version $\mathrm{sign}_P(in)$ of its input in, this protocol prefix can be used as a signing oracle. If the adversary were not allowed to define the inputs, this oracle would not be accessible and flaws might remain undetected. The adversary should also be able to determine which player plays which role in a protocol. E.g., a player named "A" should be able to first act as a contract signer and then as a notary.

Note that the fact that the adversary was unable to drive players using derived inputs and create own agents caused the insecurity in the example in Section 3 to go undetected. Otherwise, the adversary would have been able to re-send a new agent and message.

Observing Protocol Runs and Players The capabilities of an adversary largely depend on the network model. There are basically three orthogonal properties of networks:

Authentic An adversary cannot send messages on behalf of other correct parties.

Private The adversary does not obtain knowledge about the content of messages that are transmitted between correct parties.

Reliable The adversary cannot delete messages that are transmitted between correct parties.

The subsets of these properties define a half-order of network models where the model providing all those guarantees is the strongest. Furthermore, certain network connections may only be available at certain times (e.g., a private network for key exchange should disappear or be blocked after initialization). In a formal analysis, one needs to be very careful to chose the right network model. If, for example, the protocol assumes the weakest model while the evaluation models a stronger one, the verification will succeed even though the protocol may fail in practice.

For a formal analysis, the consequences are that an adversary can read any non-privacy traffic, can pretend to have sent any non-authentic traffic (e.g., pretending that a message was sent by a correct player A), and can delete any non-reliable traffic.

4.3 Deriving Knowledge

Deriving new messages or knowledge from a given set of observations is the strength of the Dolev-Yao model. It rigorously defines which messages the adversary can analyze and create by means of derivation rules, based on the set of observed messages.

One usually assumes that an adversary obtains a set of initial knowledge. This can include type IDs and the identifiers and public keys of all participants.

Splitting and Re-Assembling Messages The first step is to split messages. Splitting messages generally means decomposing tuples while learning their atoms, detaching signatures while learning contents and signerID. It should include decomposing network transmissions to learn sender and recipientID. Another safeguard is to assume that the adversary learns the encrypterID and its public key from a ciphertext.

From a cryptographic point of view, a suitable model should also include the composition and decomposition of binary strings: If a part of a secret message is included as a part of another binary string, this can clearly be a problem but it will not be detected by most models.

The adversary should be able to re-compose tuples, typed data structures, network messages, attach signatures for all non-correct parties (either initially known or unknown), and encrypt messages with any known piece of data. Examples how this can be used is to re-type a message by first detaching the type and then attaching another type (unless signed since a signature cannot be re-attached).

Polynomial Computations In current Dolev-Yao-based models, the adversary is not allowed to perform any arithmetic computation, e.g., compute $a + b$ if a and b are known. The reason is that this would enable the adversary to compute the whole message space and then non-deterministically pick a message that can be used to mount a successful attack. The solution could be to adopt the cryptographic notion of a *polynomially bounded adversary*, i.e., to only consider a polynomial number of transition while each transition can be computed in polynomial time. Since this requires a concise treatment of polynomiality, the incorporation of this topic in the Dolev-Yao model is still in its infancy. However, some important results already exist which point out that achieving this goal is indeed feasible [29, 30].

Complicated Derivations Sometimes, it would also be desirable to have certain derivations rules for capturing possible number-theoretic capabilities also in the abstract framework. For instance in RSA, it is well-known that an attack can be successfully mounted provided that two ciphertexts stand in a known linear relation. In formulas, if for a public key exponent e and an RSA modulus n two encryptions

$$c_1 = m^e \bmod n, \quad c_2 = (a \cdot m + b)^e \bmod n$$

are given where a and b are known, then m can be computed in time $O(e \log^2 e)$. Heather and Schneider tried in [22] to extend a Dolev-Yao-based attacker model such that it captures this cryptographic attack. They have shown why achieving this task is not easily possible, and they agreed with our opinion that more work is needed on adapting the attacker capabilities to cover cryptographic attacks.

4.4 On Different Attacks

We now describe actual attacks that should be detectable. Note that by controlling the players, the adversary can start any number of parallel and interleaved runs with any number of participants.

Looking for Violations of Common Robustness Principles Several work has been devoted to propose *robustness principles* for improving the design quality of cryptographic protocols [5, 3, 10]. This is done by means of describing design guidelines that should help to prevent common attacks. It is important to note that none of these principles is necessary to obtain the desired security requirements. Two of the most common principles are

- *Name the Participants:* The participant names of a protocol should be included securely in each run. A violation of this principle was the reason why the Needham-Schroeder Public Key protocol could be successfully attacked.
- *Prevent Oracles:* Make sure that parts of a protocol run cannot be used as oracles to sign or decrypt messages. Violating this principle was the reason why the attack in Section 3 was successful.

A suitable tool should check if a protocol has a flaw that violates one of the most common robustness principle. Moreover, it would even be helpful to detect if a protocol violates a principles *without* checking if this violation gives rise to an attack. The advantage is that detecting a violation is much simpler to achieve in formal proof tools than mounting the actual attack, and the result will be sufficient to point to a possible weak spot of the protocol. This weakness can then further be investigated by the protocol designer.

Attacking Synchronous Protocols A synchronous protocol assumes a global notion of rounds where all participants perform their state transitions simultaneously if they are intended to switch in the same round. Messages on reliable channels are transmitted between two rounds. A simple adversary observes all outputs and then creates inputs that are expected on unreliable channels.

However, the model may also allow for adversaries that messes with the participants during a round. E.g., it may first switch participant P_1, then using the output of P_1 as an input to switching P_2 in the same round. I.e., the adversary can interactively determine the switching sequence (within the same round) of all correct machines while selecting the (unreliable) inputs for each.

5 Conclusion

We have challenged the ability of the Dolev-Yao model, the de facto standard in verifying security protocols using tool support, to yield complete proof of security. We have substantiated our provisos by means of a protocol which has been formally verified in the Dolev-Yao model, but that has later fallen prey to an uncovered attack. In contrast to yielding proofs of security, we believe that the model is perfectly suited for detecting flaws, i.e., to increase confidence that a protocol is secure. Moreover, experience shows that proofs done within the model also provide a significant insight in a protocol's possible weaknesses. This also forces the designers to really specify all details and a precise model of the requirements that shall be satisfied. In order to increase the effectiveness of this process, we have concluded with a wish list of adversary capabilities that we believe would be desirable to capture in future Dolev-Yao extensions. The wish list is mainly motivated by the cryptographic point of view. It is surely not all-embracing and already partially realized by existing Dolev-Yao-like approaches, hence extensions of it are surely worth to be discussed.

We have challenged the ability of the Dolev-Yao model, the de facto standard in verifying security protocols using tool support, to yield complete proof of security. We have substantiated our claim by means of a protocol which has been formally verified in a Dolev-Yao-based model, but that has later fallen prey to an uncovered attack.

We nevertheless believe that the model is useful for detecting flaws. Moreover, experience shows that proofs done within the model also provide a significant insight in a protocol's possible weaknesses. This also forces the designers to really specify all details and a precise model of the requirements that shall be satisfied.

In order to increase the effectiveness of this process, we have initiated the evolution of an "adversary capability catalog" that formalizes known attacks. The larger this catalog will be the more attacks will be detected, and the higher the security assurance of the resulting formal analysis will be.

In the future, one should be able to identify or design flawed protocols that are vulnerable to exactly one of each adversary capabilities. Like the current suite of flawed authentication protocols [16], these protocols can then be used to benchmark the detection rate of a particular formal methodology.

6 Acknowledgments

We thank André Adelsbach, Birgit Pfitzmann, William Simmonds, Michael Waidner, and Markus Wittwer for valuable discussions that triggered us to write this paper.

References

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.

[2] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Proc. 4th International Symposium on Theoretical Aspects of Computer Software (TACS)*, pages 82–94, 2001.

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

[4] M. Abadi and P. Rogaway. Reconciling two views of cryptography: The computational soundness of formal encryption. In *Proc. 1st IFIP International Conference on Theoretical Computer Science*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2000.

[5] R. Anderson and R. Needham. Robustness principles for public key protocols. In *Advances in Cryptology: CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 1995.

[6] G. Ateniese, M. Steiner, and G. Tsudik. Authenticated group key agreement and friends. In *Proc. 5th ACM Conference on Computer and Communications Security*, pages 17–26, 1998.

[7] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. IACR Cryptology ePrint Archive 2003/015, Jan. 2003. `http://eprint.iacr.org/`.

[8] D. Beaver. Secure multiparty protocols and zero knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[9] F. Blanqui. An Isabelle formalization of protocol-independent secrecy with an application to e-commerce, 2002. Manuscript, available from `http://www.lri.fr/~blanqui/papers/sub02.ps.gz`.

[10] S. Brackin. Automatically detecting most vulnerabilities in cryptographic protocol analysis. In *Proc. 2000 DARPA Information Survivability Conference and Exposition (DISCEX)*, pages 222–236, 2000.

[11] J. Bryans and S. Schneider. CSP, PVS, and a recursive authentication protocol. In *Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997. `http://dimacs.rutgers.edu/Workshops/Security/`.

[12] J. Bull and D. Otway. The authentication protocol. Technical Report DRA/CIS3/PROJ/CORBA/SC/1/CSM/436-04/03, Defence Research Agency, 1997.

[13] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 3(1):143–202, 2000.

[14] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.

[15] R. Canetti, R. Gennaro, A. Herzberg, and D. Naor. Proactive security: Long-term protection against break-ins. *RSA Laboratories' CryptoBytes*, 3(1):1–8, 1997.

[16] J. Clark and J. Jacob. A survey of authentication protocol literature. *Communications of the ACM*, 21(12):993–998, 1978.

[17] Z. Dang and R. Kemmerer. Using the ASTRAL model checker for cryptographic protocol analysis. In *Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997. `http://dimacs.rutgers.edu/Workshops/Security/`.

[18] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[19] B. Dutertre and S. Schneider. Using a PVS embedding of CSP to verify authentication protocols. In *Proc. International Conference on Theorem Proving in Higher Order Logics (TPHOL)*, volume 1275 of *Lecture Notes in Computer Science*, pages 121–136. Springer, 1997.

[20] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology: CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93. Springer, 1990.

[21] J. D. Guttman, F. J. Thayer Fabrega, and L. Zuck. The faithfulness of abstract protocol analysis: Message authentication. In *Proc. 8th ACM Conference on Computer and Communications Security*, pages 186–195, 2001.

[22] J. Heather and S. Schneider. Equal to the task? In *Proc. 7th European Symposium on Research in Computer Security (ESORICS)*, volume 2502 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2002.

[23] A. Herzberg, M. Jakobson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *Proc. 4th ACM Conference on Computer and Communications Security*, pages 100–110, 1997.

[24] M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000.

[25] G. Karjoth, N. Asokan, and G. Gülcü. Protecting the computation results of free-roaming agents. In *Proc. 2nd International Conference on Mobile Agents*, volume 1477 of *Lecture Notes in Computer Science*, pages 195–207. Springer, 1998.

[26] R. Kemmerer. Analyzing encryption protocols using formal verification techniques. *IEEE Journal on Selected Areas in Communications*, 7(4):448–457, 1989.

[27] R. Küsters. On the decidability of cryptographic protocols with open-ended data structures. In *Proc. 13th International Conference on Concurrency Theory (CONCUR)*, volume 2421 of *Lecture Notes in Computer Science*, pages 515–530. Springer, 2002.

[28] P. Laud. Semantics and program analysis of computationally secure information flow. In *Proc. 10th European Symposium on Programming (ESOP)*, pages 77–91, 2001.

[29] P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proc. 5th ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[30] P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time equivalence and security analysis. In *Proc. 8th Symposium on Formal Methods Europe (FME 1999)*, volume 1708 of *Lecture Notes in Computer Science*, pages 776–793. Springer, 1999.

[31] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. 2nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.

[32] C. Meadows. Using narrowing in the analysis of key management protocols. In *Proc. 10th IEEE Symposium on Security & Privacy*, pages 138–147, 1989.

[33] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proc. 2000 DARPA Information Survivability Conference and Exposition (DISCEX)*, pages 237–250, 2000.

[34] S. Micali and P. Rogaway. Secure computation. In *Advances in Cryptology: CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer, 1991.

[35] J. K. Millen. The interrogator: A tool for cryptographic protocol security. In *Proc. 5th IEEE Symposium on Security & Privacy*, pages 134–141, 1984.

[36] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murϕ. In *Proc. 18th IEEE Symposium on Security & Privacy*, pages 141–151, 1997.

[37] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 12(21):993–999, 1978.

[38] T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

[39] L. Paulson. Mechanized proofs for a recursive authentication protocol. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 84–95, 1997.

[40] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Cryptology*, 6(1):85–128, 1998.

[41] B. Pfitzmann. Vergleich der algebraischen und kryptographischen modellierung von kryptoprotokollen. Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe, 1988.

[42] B. Pfitzmann, M. Schunter, and M. Waidner. Cryptographic security of reactive systems. Presented at the DERA/RHUL Workshop on Secure Architectures and Information Flow, Electronic Notes in Theoretical Computer Science (ENTCS), March 2000. `http://www.elsevier.nl/cas/tree/store/tcs/free/noncas/pc/menu.htm`.

[43] B. Pfitzmann, M. Schunter, and M. Waidner. Secure reactive systems. Research Report RZ 3206, IBM Research, 2000.

[44] B. Pfitzmann and M. Waidner. Attacks on protocols for server-aided rsa computation. In *Advances in Cryptology: EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 153–162. Springer, 1992.

[45] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. 22nd IEEE Symposium on Security & Privacy*, pages 184–200, 2001.

[46] V. Roth. On the robustness of some cryptographic protocols for mobile agent protection. In *Proc. 5th International Conference on Mobile Agents*, volume 2240 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2001.

[47] F. J. Thayer Fabrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. 19th IEEE Symposium on Security & Privacy*, pages 160–171, 1998.

[48] H. I. Tsutomu Matsumoto, Koki Kato. Speeding up secret computations with insecure auxiliary devices. In *Advances in Crptology: CRYPT '88*, volume 403 of *Lecture Notes in Computer Science*, pages 497–506. Springer, 1988.

[49] D. Volpano and G. Smith. Verifying secrets and relative secrecy. In *Proc. 27th Symposium on Principles of Programming Languages (POPL)*, pages 268–276, 2000.