Weakening the Dolev-Yao Model Through Probability

Riccardo Bresciani
Foundations and Methods Group
Trinity College Dublin
Dublin, Ireland
bresciar@cs.tcd.ie

Andrew Butterfield
Foundations and Methods Group
Trinity College Dublin
Dublin, Ireland
Andrew.Butterfield@cs.tcd.ie

ABSTRACT

The Dolev-Yao model has been widely used in protocol verification and has been implemented in many protocol verifiers. There are strong assumptions underlying this model, such as perfect cryptography: the aim of the present work is to propose an approach to weaken this hypothesis, by means of probabilistic considerations on the strength of cryptographic functions. Such an approach may effectively be implemented in actual protocol verifiers. The Yahalom protocol is used as an easy example to show this approach.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol Verification*

General Terms

Protocol Verification

1. INTRODUCTION

Nowadays protocol verification is mainstream in computer science. Adequate tool support has been a key aspect that has brought this about, as it made protocol verification techniques available to the protocol development community.

Protocol verification can be approached from a computational point of view, trying to take into account the complexity of the algorithms used in cryptography: this borrows ideas from complexity theory and tries to estimate from a quantitative point of view the strength of a protocol.

Another possible approach to the problem is represented by formal methods. This verifies the overall correctness of the protocol, without taking into account computational aspects. A well-known example of weaknesses uncovered by using formal methods is the Needham-Schroeder protocol: dating back to 1978, it was considered a secure protocol until 1995, when Lowe discovered an attack by using formal methods — now the corrected version of the protocol is known as Needham-Schroeder-Lowe protocol [11, 10].

SIN'09, October 6–10, 2009, North Cyprus, Turkey.

The latter approach has been mechanized by means of a variety of tools, such as ProVerif [1, 7] or AVISPA just to name a few.

Research is slowly moving in the direction of merging the two approaches, introducing some probabilistic and computational considerations to evaluate how likely it is for an attack to be performed successfully. The aim of the present work is to propose a methodology to exploit currently available tools to evaluate the probability of a successful attack on a protocol.

2. THE DOLEV-YAO MODEL

The Dolev-Yao model (sketched in figure 1) dates back to 1983 and it is still widely used in protocol verification [8]. In this model the net is seen as a star, where the attacker is in the central node: in this way every communication is mediated by the attacker, who can decide to twist the messages, to prevent them from getting to their intended recipient or to deliver them unchanged.

What the attacker cannot do is to break cryptographic functions, as they are assumed to be perfect. This means that a cyphered message can be decrypted only with the appropriate key or that a hashing function is collision-free.

The size of messages, keys and of any other term is irrelevant: this allows us to reason about the protocol abstracting from the actual implementations, as issues like overflows, different strength of keys, channel capacity and so on are not taken into account. This is useful, because the flaws that may eventually be found depend on the protocol, and not on the particular implementation of it.

Finally in the run of the protocol any number of participants is admitted: this means that there can be any number of parallel sessions that may interact.

3. VARIATIONS ON THE MODEL

The scenario depicted in the Dolev-Yao model is a highly idealized one: it can be effectively implemented in protocol verifiers, but it is somehow far from the reality of things, as it relies on strong assumptions.

For this reason variations on the Dolev-Yao model have been proposed: for example a transition system with computation times has been proposed in [6]. In this transition system each transition stands for a computation and is labeled with a function that relates the computation with its cost: the weight function returns a non-negative real number or infinity (in case of impossible computations). The use of this framework allows us to evaluate the feasibility of an attack. Later on in [6] the assumption that computation

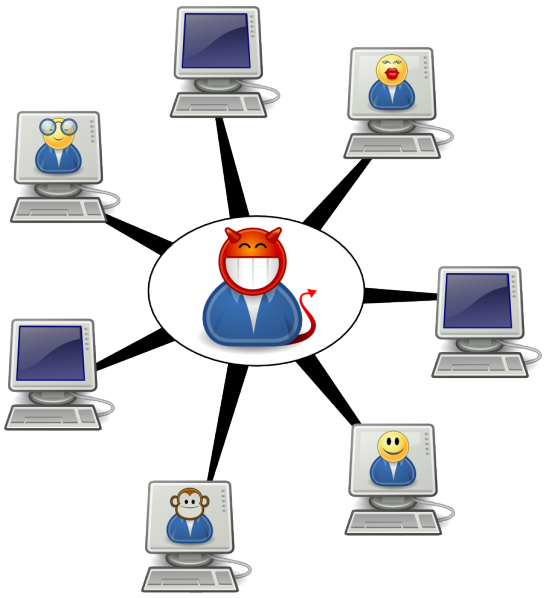

Figure 1: The Dolev-Yao model.

times are deterministic is removed and we have probabilistic computation times that label the transitions.

A probabilistic polynomial time process calculus has been proposed in [13]. In this process calculus, very similar to the π-calculus by Milner [12] and to the SPI-calculus by Gordon and Abadi [4], polynomials are associated to terms: as the aim of this process calculus is to account for probabilistic polynomial-time adversaries, this calculus must be able to express some information about the width of a channel or about the number of feasible replications of a process. This results in a complicated semantics.

Similar ideas are expressed in [14, 15], where the authors discuss the setting that results from having a probabilistic Dolev-Yao attacker who is able to guess a key with a given probability, and the related transition system. Generally speaking, the inference rules that an attacker can use are weighed by the probability that their application will be successful: the rules that characterise a classic Dolev-Yao attacker are weighed by probability $p = 1$, while the rules characterising a probabilistic Dolev-Yao attacker are weighed by a probability $p \leq 1$.

When protocol verifiers are required to prove the security of a protocol, they either state that the protocol is safe or they return a counter-example, *i.e.* the trace of an attack. This is not enough to use these tools to fully-automatically evaluate the probability of a protocol to be broken by an attack, as this would require to explore a portion of the execution tree of the possible attacks, which is larger than the one explored just to give a counter-example (when a counter-example is found, the search is simply stopped); obviously these tools can be improved to do this, exploiting for example the considerations in subsection 4.3.

In the next section an approach is proposed, in order to use actual protocol verifiers to evaluate the probability of an attack to succeed, giving in the worst case an upper and a lower bound. This approach avoids the complexity that underlies the quantitative considerations in [13], as the number of replications of processes is not limited by a polynomial and the information on channel width is hidden in the probabilistic assumptions on the strength of cryptographic functions (see the next section 4). Moreover this approach can easily be used to take into account all the extensions proposed in [14, 15].

4. A STRATEGY TO EVALUATE THE PROBABILITY OF SUCCESSFUL ATTACKS BY MEANS OF PROTOCOL VERIFIERS

This work aims at relaxing the hypothesis underlying the Dolev-Yao model: in particular we aim at reasoning in a setting where the perfect cryptography assumption is weakened.

To remove the hypothesis that cryptographic primitives are perfect means that it is possible for a one-way function to be inverted, thus revealing its argument.

If we are dealing with a sound cryptosystem, the probability of this to happen is negligible, though non-zero. It is useful to evaluate this probability: a trivial application is to estimate the security which is gained by using larger keys. An evaluation of the strength of these functions implicitly carries information about the channel width, which the calculus proposed in [13] explicitly accounts for.

We can think of two extreme cases when having to violate a cryptosystem: one option is a completely random guess, the other is collecting enough data that can be used to break cryptography. The general case is somewhere in between, ranging between these extremes.

Protocol verifiers such as ProVerif apply inference rules to derive terms from the data exchanged among the agents. The hypothesis of perfect cryptography may be instantiated in a protocol model by providing a constructor function, but no destructor function: once a constructor function is applied to some arguments it will not be possible to reverse it and find their values.

This is the case for example of hashing functions: from the cryptographic hash of a string it is not possible to recover the original string.

In the case of public key cryptography the perfect cryptography assumption is rendered as a couple of constructor-destructor functions: the destructor function will return a result only if the appropriate key is provided.

When testing the model of a safe protocol, the protocol verifier will state that no attack on the protocol is possible. Conversely if the protocol was not safe, the protocol verifier will return the trace of a possible attack.

4.1 Introducing new destructors

Starting from a safe model of the protocol, what we are aiming at is a suitable modification of the model, that accounts for a possible break in the cryptography.

We do so by inserting new destructors: if the security of the protocol was relying on the perfect cryptography assumption, this will enable the protocol verifier to find an attack on the protocol.

The naive destructors that can be added are the ones accounting for random guesses: they simply behave as oracles that invert the one-way function. For example we can provide the attacker with a destructor returning the key that is used to encrypt a message, as well as providing him with

a destructor that recovers the plain text from an encrypted message.

Finer destructors can be added, which take more than one argument. For example this may be used to take into account cryptoanalysis attacks: we can imagine a destructor enabling the attacker to recover a key after having collected a certain number of messages cyphered under that key. More in general we can model different information leaks and add destructor exploiting those leaks.

4.2 Examining the trace of the attack

We can think of these new destructors as functions that can be used by paying a price, and the price is that the probability of success of an attack is diminished accordingly to the probability that the functions used to perform that attack will return a correct value.

For example if an attack has been found and it does not use those destructors, it will succeed with probability $p = 1$: each inference rule which is applied gives a result with that probability, so the product of all those probabilities is $p = 1$.

Conversely an attack which uses once only one of such destructors will succeed with the probability that the destructor works properly: only one inference rule will succeed with probability $p \leq 1$, so the final probability of a successful attack coincides with that value.

Obviously in the case that an attack needs to use more destructors, its probability of success will be the probability that all the destructors return a correct result.

4.3 Considerations on the most successful attack

Unluckily once we have evaluated the probability for the discovered attack to succeed, still we cannot trivially be sure that there is not another attack that can succeed with a higher probability. But at least it is possible to give upper and lower bounds for the success probability, as well as an upper bound to the number of times when the attacker will have to rely on the new destructors.

The success probability p of breaking a protocol with the most successful attack is greater or equal to the success probability $\hat{p}$ of the discovered attack.

$$p \geq \hat{p}$$

Similarly we can also see that the success probability p cannot be greater than the probability p_{med} of the most effective destructor to succeed, *i.e.* the one which most likely will return a correct result: as the protocol was safe before adding the new destructors, the most favourable situation for the attacker is when he needs to apply once only the most effective destructor, as the success probability of the attack coincides with the success probability of the destructor.

$$p \leq p_{\text{med}}$$

If this is not the case, we can at least estimate what the maximum number of application of these destructors is: this is the number of times that the most effective destructor can be applied before the probability of success drops below the success probability of the discovered attack.

These considerations can be used to guide the research for a more successful attack.

5. AN EXAMPLE: USING PROVERIF TO VERIFY THE YAHALOM PROTOCOL

To illustrate the propose methodology, we will reason about the Yahalom protocol: the protocol verifier that will be used is ProVerif. It comes along with some examples files, among which there is a model of the Yahalom protocol (coded as a sequence of Horn clauses), which will be modified to suit our needs.

5.1 ProVerif

First of all a brief description of the tool used in this example.

ProVerif is a protocol verifier written by Bruno Blanchet [1, 7]. The tool processes input files formatted as a sequence of Horn clauses or as a process in the applied π-calculus (a cryptographically-oriented variation of the π-calculus), which will be translated into Horn clauses before being run.

In particular, by means of ProVerif it is possible to verify secrecy properties, *i.e.* whether a Dolev-Yao attacker is able to derive a term from the messages exchanged among the agents: for example this can be used to prove the correctness of a key agreement protocol, by proving that a term, encrypted under the negotiated key and sent on a public channel, is not derivable by an attacker. This is the way we will use ProVerif to test the Yahalom protocol.

5.2 Protocol Description

During the protocol flow (see figure 2) a total of 4 messages are exchanged among two agents, $\mathcal{A}$ and $\mathcal{B}$, and a server, $\mathcal{S}$. Both agents share a key with the server and can generate fresh nonces and keys. The server knows the identity of the agents and its own identity is publicly known. One agent knows the identity of the other agent (the one he wants to send a message to), but not viceversa.

Here are the contents of the exchanged messages:

$\boxed{\mathcal{A} \to \mathcal{B}}$ the identity of $\mathcal{A}$, together with a fresh nonce $N_{\mathcal{A}}$, is sent to $\mathcal{B}$;

$\boxed{\mathcal{B} \to \mathcal{S}}$ the identity of $\mathcal{B}$ and a triplet (made of a fresh nonce $N_{\mathcal{B}}$, the nonce $N_{\mathcal{A}}$ and the identity of $\mathcal{A}$) encrypted under the key $K_{\mathcal{BS}}$ is sent to $\mathcal{S}$;

$\boxed{\mathcal{S} \to \mathcal{A}}$ the fresh key $K_{\mathcal{AB}}$, the identity of $\mathcal{B}$ and the nonces $N_{\mathcal{A}}$ and $N_{\mathcal{B}}$, all encrypted under the key $K_{\mathcal{AS}}$, along with the couple made by the identity of $\mathcal{A}$ and the key $K_{\mathcal{AB}}$, encrypted under the key $K_{\mathcal{BS}}$, are sent to $\mathcal{A}$;

$\boxed{\mathcal{A} \to \mathcal{B}}$ the couple made by the identity of $\mathcal{A}$ and the key $K_{\mathcal{AB}}$, encrypted under the key $K_{\mathcal{BS}}$, and the nonce $N_{\mathcal{B}}$, encrypted under the key $K_{\mathcal{AB}}$, are sent to $\mathcal{B}$.

5.3 Adding Destructors to Break the Protocol

When analysing the protocol, ProVerif proves that it is safe in the classical Dolev-Yao model.

Let us now weaken the model by providing the attacker with some useful destructors, that will enable him to break the protocol: ProVerif will find that the protocol is not safe anymore.

The challenge for the adversary is to be able to decrypt the first message $M_{K_{\mathcal{AB}}}$, which is sent encrypted under the fresh key $K_{\mathcal{AB}}$.

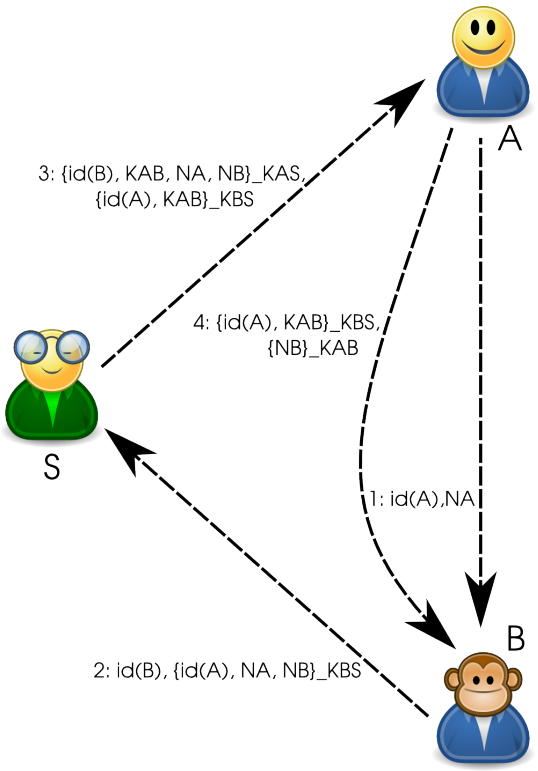

Figure 2: Message flow in the Yahalom protocol.

5.3.1 A Destructor to Guess the Key

As a first option we can imagine that the attacker can guess the key K used to encrypt a message with probability p_G. If no other knowledge about the key is available, we have that $p_G = 2^{-\ell_K}$, where ℓ_K is the length of the key K.

The attack is trivial:

- the attacker can decrypt the message $M_{K_{\mathcal{AB}}}$ if he knows $K_{\mathcal{AB}}$;
- the key $K_{\mathcal{AB}}$ can be guessed with probability p_G.

ProVerif gives the following attack trace instead:

- the attacker can decrypt the message $M_{K_{\mathcal{AB}}}$ if he knows $K_{\mathcal{AB}}$;
- the key $K_{\mathcal{AB}}$ is sent from $\mathcal{S}$ to $\mathcal{A}$ in a message which is encrypted under the key $K_{\mathcal{AS}}$;
- the key $K_{\mathcal{AS}}$ can be guessed with probability p_G.

In both cases the probability of this attack to succeed is therefore p_G, as the new destructor is used only once.

5.3.2 A Destructor to Decrypt a Message without Knowing the Key

As another option we can imagine that the attacker can decrypt an encrypted message M_K without knowing the key K with probability p_D. If no other knowledge about the contents M of the message is available, we have that $p_D = 2^{-\ell_M}$, where ℓ_M is the length of the plaintext of the message.

The attack is trivial and is the one found by ProVerif:

- the attacker can decrypt the message $M_{K_{\mathcal{AB}}}$ with probability p_D.

The probability of this attack to succeed is therefore p_D, as also in this case the new destructor is used only once.

5.3.3 A Destructor to Spoof the Server Identity

Yet another possible scenario is the one when the attacker can successfully pretend to be the server: this means that he is able to recreate the server authentication credentials. The probability of being able to do this is p_S. If no other knowledge about the private key P used to generate the server credentials is available, we have that $p_S = 2^{-\ell_P}$, where ℓ_P is the length of the key P.

ProVerif gives the following attack trace:

- the attacker breaks the server credentials with probability p_S;
- the attacker can successfully mount a *Man-in-the-Middle* attack.

The probability of this attack to succeed is therefore p_S, as again the new destructor is used only once.

5.4 Considerations on the Attack Traces

The attack traces found by ProVerif show that the Yahalom protocol is not fully safe anymore if we weaken the perfect encryption hypothesis, but it can be broken with probability

$$p = \max\{p_G, p_D, p_S\}$$

If the attacker can use all of the three destructors described above, instead of only one of them at once, ProVerif will output a trace which is similar to one of the attacks described above (actually the one using the message decryption without knowing the key), as in each one of them the new destructor is used only once, so there can be no more effective combination of using the destructors.

It must be noted that the trace given by ProVerif in this case may not correspond to the trace of the most successful attack — ProVerif knows nothing about probabilities, as this is something that we add *a posteriori*.

It must also be noted that the traces given by ProVerif are not always the most trivial ones.

6. CONCLUSION

In this paper a methodology is presented, which allows probabilistic reasoning on communications protocols, by means of existing tools — provided that their output is an attack trace, in the case that an attack is feasible.

The classic setting depicted in the Dolev-Yao model is extended to a broader one, where a probabilistic Dolev-Yao attacker can perform actions that are not compatible with the perfect cryptography assumption: these actions may succeed with a given probability.

The advantage of this methodology is that it does not require any modification to be apported to the tool used, as all of the probabilistic reasoning is made on the output of the tool that is used.

The disadvantage is that the output of the protocol verifier is not always enough to find the most effective attack that can be carried out on the protocol: ideally we would like to have the protocol verifier to output the most effective

attack, but this cannot happen as the effectiveness of an attack depends on the probabilistic assumptions that are made and that therefore do not affect the analysis carried out by the tool.

Nonetheless this is enough to determine an upper and a lower bound for the probability of the protocol to be broken: these bounds depend on the likelihood that the probabilistic Dolev-Yao attacker may successfully perform the operations that we entitle him to do, operations that a classic Dolev-Yao attacker is not allowed to do.

Future work will be in the direction of integrating this methodology in one of the available tools, by making it aware of the weight of inference rules: by doing so the resolution algorithm may be modified and applied to explore different portions of the feasible attacks tree, in order to find the most effective one.

7. ACKNOWLEDGMENTS

The present work has emanated from research conducted with the financial support of Science Foundation Ireland.

8. REFERENCES

[1] M. Abadi and B. Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. *Journal of the ACM*, 52(1):102–146, Jan. 2005.

[2] M. Abadi, B. Blanchet, and H. Comon-Lundh. Models and Proofs of Protocol Security: A Progress Report. In *21st International Conference on Computer Aided Verification (CAV'09)*, Lecture Notes on Computer Science, Grenoble, France, July 2009. Springer Verlag. To appear.

[3] M. Abadi and C. Fournet. Mobile Values, New Names, and Secure Communication. In *POPL '01: Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 104–115, New York, NY, USA, 2001. ACM.

[4] M. Abadi and A. D. Gordon. A Calculus for Cryptographic Protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[5] M. Abadi and P. Rogaway. Reconciling Two Views of Cryptography (the Computational Soundness of Formal Encryption). *J. Cryptology*, 15(2):103–127, 2002.

[6] M. Baudet. Random Polynomial-time Attacks and Dolev-Yao Models. *Journal of Automata, Languages and Combinatorics*, 11(1):7–21, 2006.

[7] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop*, pages 86–100, 2001.

[8] D. Dolev and A. C. Yao. On the Security of Public-Key Protocols. *IEEE Transaction on Information Theory*, 2(29):198–208, March 1983.

[9] O. Goldreich. *Modern Cryptography, Probabilistic Proofs, and Pseudorandomness*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[10] G. Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Inf. Process. Lett.*, 56(3):131–133, 1995.

[11] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR. In *TACAs '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 147–166, London, UK, 1996. Springer-Verlag.

[12] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, June 1999.

[13] J. C. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A Probabilistic Polynomial-Time Calculus for Analysis of Cryptographic Protocols. In *Electronic Notes in Theoretical Computer Science*, 2001.

[14] R. Zunino and P. Degano. A Note on the Perfect Encryption Assumption in a Process Calculus. In *FoSSaCS*, pages 514–528, 2004.

[15] R. Zunino and P. Degano. Weakening the Perfect Encryption Assumption in Dolev-Yao Adversaries. *Theor. Comput. Sci.*, 340(1):154–178, 2005.