How to explain security protocols to your children

Véronique Cortier[1] and Itsaka Rakotonirina[2]

[1] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
[2] MPI-SP, D-44799 Bochum, Germany

Abstract. Security protocols combine two key components: a logical structure (who answers what, under which conditions?) as well as cryptography (encryption, signature, hash, ...). It is not so easy to explain their principles and weaknesses to a non expert audience. Why is something an attack or not? For which attacker? With what purpose?
In this paper, we propose an approach to introduce security protocols to a general audience, including children or even scientists from different fields. Its goal is to convey the implicit assumptions of our community, such as threat models or the participants' behaviour. This all-public introduction can be thought of as a story but, interestingly, can also be implemented physically with boxes and padlocks: manipulation helps to understand how protocols operate, even permitting non-expert participants to design their own—and thus to size the challenges of this task.

This paper is dedicated to Joshua Guttman, on the occasion of his 66,66 birthday, with many thanks for his inspiring conversations.

1 Introduction

How do you explain security protocols to your children? You would typically start with applications. "You see, security protocols are very useful. They are used in payment, 5G, in your messaging applications, biometric passport, and even voting." Yes, but how does it work? And here starts the complex part. You would need to explain a bit of *cryptography*, the expected behaviour of the participants, and how attackers can deviate from them. When giving examples of *attacks*—that is, the operating modes of malicious parties breaching the protection offered to other users—we get recurring questions: but Bob can clearly see that he has been tricked into sending Alice's key in clear! Why doesn't he stop?

This paper describes a fictitious situation where Isabelle and Bob wish to exchange a cake while preventing the deliveryman from eating it. To this end, boxes and padlocks are used, acting as abstractions of various flavours of data encryption. The deliveryman thus represents the standard threat model in security protocols: an attacker who may see or block any package circulating on the network, who may as well create and inject messages of her own, but who cannot open encrypted messages without the corresponding keys. The objective of this story is to offer a glimpse of the subtleties of security—and of the unexpected turns of events that can arise when protecting deliveries with unbreakable padlocks. An early version has appeared on a popularisation website [19] (in French). And, of course, we have used it already in many talks and activities.

Our story progressively builds a protocol, alternating between the (playful) presentation of preliminary versions and attacks on them. It stops one step before obtaining the full Needham-Schroeder protocol [17]. Interestingly, this story can also be simulated physically, as an interactive puzzle using real boxes and padlocks. Manipulation helps understanding how protocols work, even permitting to non-expert participants to design their own protocol—often to see, to their surprise, that it can easily be breached. In our experience, this type of interaction helped general audiences to understand the pitfalls and challenges of the field, compared to a purely verbal presentation.

2 Storyboard

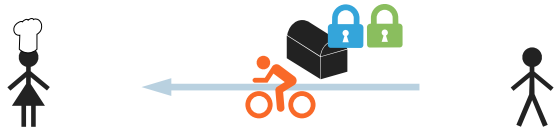
Bob would like to buy a cake from a famous bakery run by Isabelle. The store being too far away to go in person, he orders a home delivery. Unfortunately, the deliveryman is known for having a sweet tooth: he tends to compulsively eat the cakes he is supposed to deliver. It is a complex situation: Isabelle and Bob *need* the deliveryman, but they are also aware of the fact that *he cannot be trusted*. After discussing the issue over the phone, they realise that they both have *padlocks* at home: when using this unreliable postal service, they could put Isabelle's deliveries inside locked boxes to mitigate the risk of theft.

The issue is that Bob is not in possession of Isabelle's key, and thus cannot open her padlock; simply sealing the bakery's orders with it is therefore not a solution by itself. Maybe Isabelle could send the key later? But then the delivery person could get both the locked box and its key, rendering the protection ineffective.

2.1 Our first security protocol

A first solution is to combine the two padlocks as follows. Isabelle first places the cake in a box, locks it with her padlock, and sends the whole safe to Bob.

Of course, Bob is unable to open the box. Instead, he adds his own padlock and sends the box back to Isabelle. The safe is now locked by both padlocks.



Upon reception of the package, Isabelle removes her padlock and sends the box one final time to Bob.

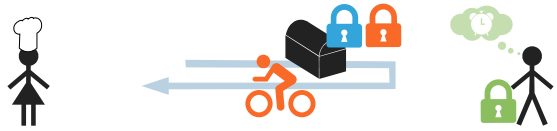
Bob can finally remove his lock and get the cake. These three steps can be followed in order to send a cake while providing protection against malicious interferences; saying it differently, this is an example of a first security protocol! Let us call it V1. No keys were sent during the process and there was always at least one lock on each package given to the delivery person. Hence, he cannot eat Isabelle's cake... or can he? This statement is actually relying on implicit assumptions:

1. the deliveryman cannot open a padlocked box, and
2. he is *passive*, that is, he follows the delivery instructions he receives.

When designing a security protocol, in particular its *logical structure*, we typically assume secure building blocks such as the soundness of cryptography. In the case of encryption, it means assuming that the attacker cannot read an encrypted message unless he has the decryption key; in our story, this is the analogue of the deliveryman not being able to open locked boxes. However, what about the second assumption?

2.2 How the postman steals Isabelle's cake

Actually, our first protocol is too weak against an *active* malicious deliveryman, that is, one who may deviate from the instructions Isabelle and Bob give him. Indeed, instead of delivering the box to Bob, he may add his own lock to the package and return it to Isabelle.



Isabelle does not know that the second padlock is not the one from Bob: the deliveryman's is just like any other. Therefore, Isabelle believes that everything is running as expected and proceeds with the next step of the protocol—that is, she removes her padlock.

The deliveryman may then unlock the box and eat the cake! This is a first example of an *attack*, against protocol V1: despite the locks, the deliveryman has a way to steal the cake. This did not require breaking any lock: it simply exploits a hole in the protocol's structure.

2.3 A fix: asymmetric encryption

Without extra assumptions about the initial setting, it is actually impossible to design a protocol secure against an active attacker. Hence, Isabelle and her clients decide to subscribe to a large certification organisation with *public padlocks*:

- Each member of the organisation is given a personal, private key;
- Everyone can get from the organisation a padlock of a specific member.

Note that the deliveryman may also be part of the organisation like anyone: let us not prevent him from buying cakes to Isabelle without stealing them! With this organisation, there is now a simple protocol (V2): Isabelle places Bob's cake in a box and seals it with Bob's padlock, that she received from the organisation. Only Bob can open his padlock hence only Bob can get the cake.

In technical terms, this corresponds to asymmetric encryption and signature: everyone can encrypt a message with Bob's public key but only Bob can decrypt. One important issue is to be certain to use Bob's public key and not another one. Imagine for example that the deliveryman was able to make Isabelle believe that his own padlock is actually Bob's: he would again gain access to Bob's cake. This explains why public keys are typically *certified* by certification bodies and why, sometimes, your browser tells you that it cannot recognise a certificate—and hence that you should not trust the corresponding website.

2.4 Denial of service?

In addition of having a sweet tooth, the deliveryman is also a sore loser: frustrated by such a simple but secure solution, he takes the locked box containing Bob's cake and throws it away instead of delivering it. In the Internet, a drop

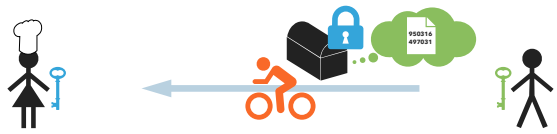
of one packet is not a real issue: the packet will be sent again as it is easy to keep a copy of it—at least it is much easier than re-baking a delicious cake. If an attacker tries to systematically block any packet sent by Isabelle, this forms what is called a *denial of service attack*. They are typically considered as out of scope of security protocols since they should be prevented by other means, at the network level; we therefore ignore them in our context.

3 When it gets really dark

But this is not the end of the story. Bob has enemies as well as friends, and someone sends him a package with a poisoned cake and a message claiming it is a gift from Isabelle. Happy, Bob eats it, which reveals a critical weakness of the protocol: we can never be sure that the sender of a delivery is who he claims to be. When Bob returns from the hospital, he and Isabelle decide to update the delivery protocol to ensure some form of *authentication*.

3.1 Challenge and respond

We assume that Isabelle and her client (here Bob) have obtained padlocks from the organisation. This time, the customer is the one initiating the protocol, writing on a piece of paper a (long) password that will be used to identify his order. He then sends it to Isabelle in a box sealed with her lock, and attaches a note with his own name so that Isabelle knows who to return the delivery to.



Upon receiving the box, Isabelle reads the customer's name on the note, opens the box with her key, puts the cake inside next to the paper with the password, and then seals it with the customer's padlock. She then has the package delivered.

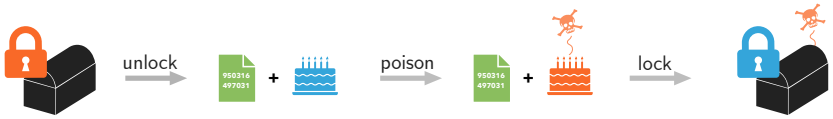
The client checks that the cake is accompanied by the same password that he chose initially; if not, he can, say, destroys the cake. We call this new version V3. It is designed so that someone attempting to make Bob eat a poisoned cake would have to guess his password—which is considered to be impossible if it is long and unpredictable enough. This is an example of a widely-used technique, called *challenge-response*. Here, Bob challenges Isabelle to answer with his long, temporary, password. Only Isabelle can open her box, hence only Isabelle can respond with the correct password. Such a challenge-response technique is used in many protocols, such as EMV [9] or biometric passports [13].

So what do you think? Is the V3 protocol secure?

3.2 Man in the middle attack

Unfortunately there is yet another attack even on our reinforced version of the protocol. This is also what makes security protocols so interesting: without formal, rigorous security analyses, we often end up with such counter-intuitively broken protocols. The actual attack unfolds as follows. First of all, the protocol starts as usual: Bob places a password in a locked box and writes his name on it. On his side, the deliveryman orders a cake, then transmits Bob's order to Isabelle *but* replaces Bob's name by his own.

Isabelle follows the instructions of the protocol and places the cake in the safe, thinking that the password inside is from the deliveryman. She then closes the box *with the deliveryman's padlock*: this permits him to open it, poison the cake, and close it afterwards with Bob's lock.



The deliveryman then returns to Bob and gives him the poisoned delivery, pretending it is what Bob ordered from Isabelle. Bob believes it to be safe due to the presence of his password and eats the poisoned cake...

3.3 A countermeasure

A countermeasure is to ask the customer to put his name *inside* the box, next to the password, to prevent the deliveryman from changing it. Actually, adding more information about identities inside boxes (or, inside ciphertexts in real protocols) is commonly considered as a good practice for similar reasons [1]. Calling this final version V4, it finally satisfies the expected property:

> *An active malicious delivery person can neither eat a cake sent to Bob, nor poison Bob if the latter only eats cakes he thinks are from Isabelle.*

Due to the attacks we exhibited on all successive versions of the protocol, you might be wary of that statement—as of any security statements you hear from today. The version V4 however deserves more trust than the previous ones, as its analysis has been backed up by formal, automated tools [18], thus mitigating the risk of unconsciously overlooking attack scenarios, or simply of inattention mistakes. Formal methods allow to *prove* that there is no attack, even when the deliveryman sends arbitrary packets, in any order. These techniques are applied more generally to security protocols to detect flaws and obtain enhanced

guarantees. Some of them are based on strand spaces [22,12,11], Horn clauses [2], or constraint systems [20]; examples of such techniques can be found in the following survey book [8].

It is also interesting to note that protocol V4 corresponds to the first part of the Needham-Schroeder protocol [17], that has received in-depth academic scrutiny, including analyses in popular automated tools such as ProVerif [3], Tamarin [16], or Maude-NPA [10]. These tools continue to receive attention and improvements from the community up to this day [5,14], in parallel to the emergence of new techniques and analysers that with yet different approaches [4,7,6]. The attack on V3 is actually very similar to an attack found against the second part of the Needham-Schroeder protocol [15], also due to missing indications of Bob's identity. The description of many other protocols and attacks can be found in the reference book from Schneier [21].

4 A practical session

These stories can also be played "in real life": under the form of an interactive riddle, the protocols and their attacks are easier to understand. This also gives the opportunity for participants to design their own protocols, and thus to estimate the difficulty of the task. We explain here how we organise a practice session; we experimented with it on audiences from diverse backgrounds, ranging over children, teenagers, teachers, or adults of various ages and professional activities—and it basically works with anyone curious enough. Sessions usually last about 45 minutes, but can be adapted to 30 minutes or stretched to 1 hour easily with minor adjustments. A good size for the group is about 10-15 participants, as they later need to be split into 2 or 3 smaller subgroups of 3-5 participants.

4.1 The material

In short, a session requires the following material, discussed below:

1. *boxes and padlocks*: ideally three should be available per subgroup;
2. *pens and post-its*, for the passwords of the challenge-response;
3. *small toys* to represent Isabelle, Bob, the deliveryman, the public-lock organisation, and the cakes.

Boxes can be crafted using cardboard paper; more advanced versions like the ones on the picture can be built with reasonable effort, using paintable wood chests (obtainable from hobby shops), where two metal rings are screwed (found in hardware shops, and through which padlocks can be closed to lock the boxes).

It can also be fun to have various sizes of boxes to allow for "Russian dolls", that is, nested boxes that simulate several layers of encryption. As this is not necessary for solving the riddles presented here, this is simply a misleading pitfall. In our experience, providing such spurious mechanisms improves the imaginativeness of the participants, and is a good way of not guiding them too much towards the solution—making them understand that stacking all available ingredients does not necessary improve security as a result.

Regarding toys, you can just borrow them from your children or nephews. We typically use characters from a world-known brand but we did not get the permission to use them in official material or pictures. You may also use chess pieces or any wood token if you do not like gendered figurines.

4.2 A typical session

Part 1: Context. In a typical session, after a short introduction about the history of security and telecommunication, we follow the storyboard, using our boxes, padlocks and figurines to play the scenarios. First of all, we thus explain V1; then, before telling the attack, we ask two volunteers to come and play the protocol's roles themselves (one plays Isabelle, one Bob). This helps making sure everyone got enough time to understand the setting. Of course, *we* play the delivery person and attack the two volunteers as soon as we feel that the group is ready for it.

After explaining the notion of public locks and letting the audience find the (easy) protocol V2, we explain the new scenario: the deliveryman now wishes to poison Bob and we should find a protocol to protect him from such a disaster.

Part 2: Design. We then split the audience in 2 or 3 smaller groups and ask them to invent their own protocol for the above purpose. Each group receives material and has about 15 minutes for this activity. In the meantime, we regularly check on them to re-explain the goals and assumptions, or to clear misunderstandings. Recurring misconceptions are typically on the following aspects of the problem:

1. *The protocol design is not secret*: as a protocol is designed for everyone to use, we assume that everyone knows how it works—including the deliveryman. We thus deny assumptions such as "the deliveryman has no way to know Isabelle and Bob plan to do this" as soon as we overhear them.
2. *The attacker is unpredictable*: assumptions such as "if I do this, the attacker will think that, and he will therefore not do this action that would break my protocol" should also be denied by the animators.
3. *The attacker is ubiquitous*: it is not possible to keep the attacker busy somewhere else to avoid his interferences—for example by sending dummy boxes to another destination.

In our experience, younger children (10-12 years old) are more optimistic and happier with their protocol. Older ones better anticipate attacks and, as a result, may be more reluctant to propose their own design; it is usually helpful to encourage them to try anything as a starting point, to be improved step by step.

Part 3: Attacks. Then comes the time when each group explains its protocol to the rest of the participants. This leads to a friendly competition where each group tries to find attacks on the others. At this point, we consider that V3 is sufficient; more generally, we do not mention attacks that require the deliveryman to have accomplices among the clients. Interestingly, it is often the case that participants easily find attacks on other groups' protocols, while they struggle to find one against their own—even when it is deeply flawed or close to identical to another group's. It is then easy to convince them that designing TLS was not so easy...

Naturally, in case nobody finds an attack on a broken proposal, this is our task to find one (which requires to be comfortable with security analyses).

Part 4: Ending. The final part is up to you, the group, and the remaining time. One possibility is to go further into the protocol design and explain the attack on V3, present V4, and link them to real protocols that use challenge-response as a buliding block. If you feel like it, you can then continue with fancier protocols (voting, payment, passport, messaging...). Another direction is of course to talk about formal methods. At this stage, the participants are quite aware of the fact that it is difficult to design a protocol, and even more difficult to be convinced that there is no attack. They will expectedly be receptive to the message that we need rigorous techniques to *prove* that there is no flaw.

4.3 Long-term variants of the design-attack parts (advanced)

Our experience is that even college students are receptive to this playful format. We therefore also investigated longer-term, more technical variants of the practical session, serving as a student project for introduction courses to security. One of the authors (Véronique Cortier), along with other teachers, has implemented it several years in a French engineering school, for the equivalent of third-year bachelor students. We outline in this section how this may be organised.

First of all, the Part 1 of the practical session (from the historical context to presenting the poison scenario) can be used as is to tease the overall course to the students. A couple of hours may then be spent in a more classical, academic manner to connect this game to practice: more technical definition of symmetric and asymmetric encryption, presentation of the Alice-Bob notation to specify protocols, or essentially any basic knowledge on security that the course is expected to convey. Once the students start having a minimal understanding of protocols and of a syntax to specify them, the actual project can be presented: it replaces the Parts 2 and 3 of the activity, and students will carry it out mostly out of class, in parallel to the course, under the form of an online competition.

1. Just as in Part 2 of the practical session, students form small groups and work on the design of protocol V4. A rudimentary online interface should allow each team to put their proposals in full view of all other students.
2. All along the project, as in Part 3, the goal of each team is to attack other team's protocols, in addition of proposing its owns. A scoring system is then designed to encourage playing the game; it should:

(a) reward finding attacks, all the more for attacks found quickly;
(b) reward proposing (new) protocols lasting long before an attack is found;
(c) *optional*: in case the course teaches the use of automated tools such as ProVerif or Tamarin, protocols coming with a formal analysis in one of these tools may be rewarded, just as attacks found using them;
(d) *advised*: influence the solutions to stay away from the V3/V4 protocols, which are usually found quickly, wasting the competition. We used a *cost function*: each operation (encrypting, decrypting, pairing...) has a determined cost, and the overall cost of a protocol induces a score penality, thus encouraging to avoid costly operations. It then suffices to smartly choose a cost function that spikes on V3/V4 and remains reasonably low otherwise. Our choice was to attribute a very high cost to encrypting a tuple of messages (which corresponds, in V3 for example, to putting two objects—the password and the cake—inside the same locked chest).

Of course, the competition should remain beneficial to the students to foster motivation and avoid despondency. We simply gave bonus points to the final exam to students depending on their rank in the competition. Dedicated grades could also be given, provided an honest investment secures a reasonable mark.

5 Conclusion: have fun!

We hope this paper and the associated activities can help explaining security protocols to a general audience and will be pursued with other ideas. At least, we had a lot of fun in all of our sessions!

Acknowledgments

We would like to thank Isabelle for her fantastic cakes at our cafeteria. It certainly fostered our motivation to work on security protocols[1].

[1] and also to take a few naps.

References

1. M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.
2. B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, June 2001.
3. B. Blanchet. Automatic verification of security protocols in the symbolic model: The verifier ProVerif. In *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*, volume 8604 of *LNCS*, pages 54–87. Springer, 2014.
4. R. Chadha, V. Cheval, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. *ACM Transactions on Computational Logic (TOCL)*, 17(4):1–32, 2016. Source and other references available at `https://github.com/akiss/akiss`.
5. V. Cheval, V. Cortier, and M. Turuani. A little more conversation, a little less action, a lot more satisfaction: Global states in proverif. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 344–358. IEEE, 2018.
6. V. Cheval, S. Kremer, and I. Rakotonirina. The deepsec prover. In *International Conference on Computer Aided Verification (CAV)*, 2018. Website: `https://deepsec-prover.github.io/`.
7. V. Cortier, A. Dallon, and S. Delaune. Sat-equiv: an efficient tool for equivalence properties. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 2017. Website: `https://projects.lsv.ens-paris-saclay.fr/satequiv/index.html`.
8. V. Cortier and S. Kremer, editors. *Formal Models and Techniques for Analyzing Security Protocols*, volume 5 of *Cryptology and Information Security Series*. IOS Press, 2011.
9. EMVCo. Book1 – application independent ICC to terminal interface requirements. Technical report, November 2011.
10. S. Escobar, C. Meadows, and J. Meseguer. A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties. *Theoretical Computer Science*, 367(1-2):162–202, 2006.
11. J. Guttman. Cryptographic protocol composition via the authentication tests. In *Foundations of Software Science and Computation Structures (FOSSACS'09)*, Lecture Notes in Computer Science, March 2009.
12. J. Guttman and F. Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 2001.
13. ICAO. Machine readable travel documents. Technical report, International Civil Aviation Organization, 2006. Doc 9303. Part 1.
14. D. Jackson, C. Cremers, K. Cohn-Gordon, and R. Sasse. Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2165–2180, 2019.
15. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, march 1996.
16. S. Meier, B. Schmidt, C. Cremers, and D. Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In N. Sharygina and H. Veith, editors,

Computer Aided Verification, 25th International Conference, CAV 2013, Princeton, USA, Proc., volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013.
17. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12):993–999, 1978.
18. I. Rakotonirina. *Efficient verification of observational equivalences of cryptographic processes : theory and practice*. PhD thesis, Université de Lorraine, 2021.
19. I. Rakotonirina. Les livraisons dangereuses, January 2021. On Interstices (in French): `https://interstices.info/les-livraisons-dangereuses/`.
20. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Computer Society Press, 2001.
21. B. Schneier. *Applied Cryptography Second Edition: protocols, algorithms, and source code in C*. J. Wiley & Sons, Inc., 1996.
22. J. Thayer, J. Herzog, and J. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. of the IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 1998.