Kent Academic Repository

Boiten, Eerke Albert (2010) *Security specification: completeness, feasibility, refinement.* In: Extended Abstracts Collection -- Refinement Based Methods for the Construction of Dependable Systems, Dagstuhl, Germany.

Downloaded from

https://kar.kent.ac.uk/30692/ The University of Kent's Academic Repository KAR

The version of record is available from

http://drops.dagstuhl.de/opus/volltexte/2010/2374/

This document version

Pre-print

DOI for this version

Licence for this version

UNSPECIFIED

Additional information

http://drops.dagstuhl.de/opus/volltexte/2010/2374

Versions of research works

Versions of Record

If this version is the version of record, it is the same as the published version available on the publisher's web site. Cite as the published version.

Author Accepted Manuscripts

If this document is identified as the Author Accepted Manuscript it is the version after peer review but before type setting, copy editing or publisher branding. Cite as Surname, Initial. (Year) 'Title of article'. To be published in *Title of Journal* , Volume and issue numbers [peer-reviewed accepted version]. Available at: DOI or URL (Accessed: date).

Enquiries

If you have questions about this document contact ResearchSupport@kent.ac.uk. Please include the URL of the record in KAR. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies).

Security specification: completeness, feasibility, refinement

Eerke Boiten

Computing Laboratory, University of Kent, Canterbury, Kent, CT2 7NF, UK.
E.A.Boiten@kent.ac.uk   www.cs.kent.ac.uk/~eab2/

Abstract. The formal methods and refinement community should be able to contribute to the specification and verification of cryptography based security protocols. This paper describes a few of the challenges that arise in this context. These include: security properties which differ from one application to another, and as a consequence issues of specification completeness; approximate rather than absolute notions of security, and underlying theories which do not provide obvious methods for "correctness by construction".

1 Introduction: Commitment and Completeness

At a first glance, cryptographic protocols provide exactly the kind of problems that formal methods are most suitable for and perform best at: short programs (most fit on a single page), based on rich algebraic mathematics, whose correctness is highly critical. However, the mathematics and the notions of security (correctness) are very different from the usual formal methods repertoire.

As an example, consider the cryptographic primitive of *bit commitment.* This is an essential ingredient of many cryptographic applications, particularly to build up trust between different parties, e.g. in authentication, and for zero-knowledge proofs of knowledge. Commitment, informally, is like putting a value in a locked box. One party (the "committer") chooses a message, and transforms it in a way which makes sure they cannot later claim it was a different message (it is "binding"), and the other party (the "receiver") cannot see it (it is "hiding"), and passes the transformed message to the other party (it "commits"). After commitment has taken place, typically further interaction will follow. At some point the commitment may be opened, e.g. as a check for honesty. One then expects the original message to be retrieved ("correctness"). The three properties: hiding, binding, and correctness together constitute the specification of commitment. When the message is a single bit, it is called *bit commitment.*

Another method of specifying this functionality is in what is commonly called the "ideal model" [12], where parties can communicate securely with an incorruptable third party. In that model, the committer sends their message to the trusted intermediary, who then confirms to the receiver that some commitment is made. When the committer asks for the commitment to be opened, the intermediary sends the original message to the receiver. Security of an implementation in this model means: any attack that succeeds against the implementation is as likely to succeed against the ideal model scenario. Although we have omitted the formal details here, it appears as if hiding, binding and correctness are indeed guaranteed in this specification. Conversely, it would be difficult to come up with a simpler "ideal model" specification that satisfies those three properties. We will come back to this in detail in Section 2.

To consider a rather more complicated example, protocols for electronic voting have been studied for many years, leading to an extensive list of desirable security properties [11]:

> fairness, eligibility, individual verifiability, universal verifiability, vote-privacy, coercion-resistance, receipt-freeness

with many of these varying depending on whether computers and election officials can be trusted or not. Even though the cited work provides a lot of structure by establishing relations between all these properties, it would still be hard to be certain that this set of properties or any future extension completely covers all possible attacks on an electronic voting protocol [9].

In general, when stated security properties closely match attacks that have been envisioned, clearly any verification is relative to the set of attacks considered, and completeness remains an issue. The cryptographic security community is undecided as to whether ideal model specification addresses this completeness problem [9, 10].

2 Commitment and Feasibility

We examine commitment and its security properties in more detail here. The context in which commitment schemes must be understood is as part of a protocol. A *protocol* involves at least two parties and is an algorithmic prescription for a number of causally related communications between the involved parties, aiming to achieve a particular objective. A protocol may succeed, or it may fail. It fails when the exchange of messages stops prematurely, for example after one party observes that another party is not adhering to the protocol. It succeeds if the protocol has completed and none of the parties has declared failure explicitly. Parties which act according to the protocol's rules and aim to achieve the protocol objective are called "honest". If the protocol succeeds although the objective has *not* been achieved, this indicates a breach of security. The protocol is *expected* to fail if some of the parties act dishonestly – thus, it is never in the interest of a dishonest party to perform an action that is *guaranteed* to lead to the protocol's failing. A practically relevant expectation is that a cryptographic protocol has a fixed number of fixed size messages, where the numbers may depend on the sizes of any protocol parameters, or on a security parameter (such as a key size).

The bit commitment scheme consists of three phases: preparation, committed, and opened. In the preparation stage, no bit has been chosen yet; in the commited stage, the committer has chosen a bit b that they cannot change (binding), and that the receiver does not know (hiding); in the opened phase, the receiver knows that the committer originally committed to b. The transitions between phases are achieved by messages from the committer to the receiver.

A first, obviously broken, attempt at a protocol is where the committer sends out a value $commit(b)$ for a known function *commit*, and later the value b as an opening. Correctness is guaranteed, but hiding normally is not: the receiver can check immediately by "exhaustive" search whether they received $commit(0)$ or $commit(1)$. However, if $commit(0) = commit(1)$ then hiding is guaranteed, but binding is not.

The normal solution for this is *randomisation*. The traditional argument for the need for randomisation in cryptography is masking known distributions within plaintexts in encryption algorithms – this is another. There are two common views of probabilistic algorithms. One view is that they include explicit probabilistic choices "inside". This model fits best when e.g. considering the combination of non-deterministic and probabilistic specification [13]. The other view is to consider "deterministic extensions" of probabilistic algorithms: these are deterministic algorithms which take an additional argument (sampled from a given distribution) representing the actual probabilistic choices made. In the context of commitment, we need the latter view. Thus we end up with a new specification, where *commit* takes an additional argument, which is also sent at opening time to allow the receiver to verify correctness. However, due to the assumption of bounded sized messages, this additional argument is bounded, and thus both sides can attempt to cheat. When the committer has sent out $c = commit(0, r)$, he can search for r' such that $commit(0, r) = commit(1, r')$ in order to defeat the binding property: he could claim to have committed to 1 and provide r' as the evidence. Thus, such r' should not exist. However, in order to defeat hiding, the receiver can search exhaustively for r such that $c = commit(0, r)$ or $c = commit(1, r)$. One of these is guaranteed to succeed, so the only case where hiding succeeds because the cheating receiver has no information is when $c = commit(0, r) = commit(1, r')$ – exactly the case where binding fails. Thus, binding and hiding are contradictory properties, and a protocol of the suggested shape satisfying both cannot exist, despite the existence of an "ideal model" specification.

However, commitment *is* considered useful, even if practical schemes cannot live up to the ideal. The compromise of completely dropping one of the two crucial properties is clearly unacceptable – and we can do significantly better than that, by bringing in a computational notion of correctness. In summary: the ideal combination of "perfect" binding and hiding is not achievable; however, the literature shows that schemes exist which approximate both as closely as required, provided we assume (dishonest) parties to be constrained to bounded (polynomial) time. We will briefly describe approximate notions of correctness and refinement in Section 3.

A final aside about commitment relates to the notion of *universal composability* [7]. This is a formal methods inspired concept, of conditions which would allow compositional reasoning, in the sense that one could substitute the ideal model specification of a scheme instead of an implementation when reasoning about protocols built using the scheme. For commitment, it has been proved that an implementation satisfying this strong compositional notion of correctness cannot exist [8, 10].

3 Computational Correctness

Notions of statistical and computational correctness ("provable security") in cryptography are built on the idea that breaking a system may only need to be hard and unlikely, rather than theoretically impossible. For commitment this is the best that can be achieved; for other applications it may be more realistic and efficient. Attack models then include explicit probabilism, reflecting situations like it always being possible to correctly guess an encryption key, though with a very small probability only. Informally, the computational version of the hiding property is as follows. Let the randomising argument to *commit* have length n bits. Then, for any probabilistic algorithm with time complexity polynomial in n, the probability of it distinguishing outputs of *commit* for bit 0 with uniformly chosen randomising input, and similar for bit 1, is negligible (i.e., smaller in the limit than 1 divided by any positive polynomial). A similar definition exists for binding, and commitment schemes have been defined in the literature that achieve one security property in the absolute sense, and the other in the computational sense described here.

For more details of this and the reconstruction of the commitment primitive from a formal methods perspective, see the draft paper [4]. Clearly there is a connection between the notion of approximate correctness described here, and our notion of approximate refinement [6] – the draft paper also gives more details of that.

4 Towards Correctness by Construction

For a slightly more extensive discussion, see [5]. The state of the art for cryptographic protocols is that verification is done post-hoc only, with very little machine support. Proofs for provable security are hard: notations and theories such as probability theory and complexity theory do not have strong algebraic traditions or properties. In particular, proofs over "all probabilistic polynomial algorithms" have no induction principles to support them, so are typically carried out by contradiction and probabilistic reduction. ("If we had an efficient algorithm to break this cryptographic scheme, this could be used to solve a known difficult number-theoretic problem.") Promising approaches in this area include universal composability discussed above, and "game hopping" [2] supported by the CryptoVerif proving system [3].

On the formal methods side, recent developments in probabilistic refinement [13, 16], action refinement [1], and secrecy-preserving refinement [15, 14] contribute to solving the problem of finding refinement relations that will one day allow us to derive cryptographic protocols from abstract specifications, providing correctness by construction.

References

1. R. Banach and G. Schellhorn. On the refinement of atomic actions. *ENTCS*, 201:3–30, 2008. Proceedings BCS-FACS Refinement Workshop 2007.
2. M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.
3. Bruno Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, October–December 2008. Special issue IEEE Symposium on Security and Privacy 2006. Electronic version available at http://doi.ieeecomputersociety.org/10.1109/TDSC.2007.1005.
4. E.A. Boiten. Commitment: A challenge for formal methods, 2008. Draft paper, www.cs.kent.ac.uk/~eab2/crypto/commit.pdf.
5. E.A. Boiten. From ABZ to cryptography (abstract). In E. Börger, M. Butler, J.P. Bowen, and P. Boca, editors, *ABZ 2008*, volume 5238 of *LNCS*, page 353. Springer, September 2008. www.cs.kent.ac.uk/~eab2/crypto/abz.pdf.
6. E.A. Boiten and J. Derrick. Formal program development with approximations. In H. Treharne, S. King, M. Henson, and S. Schneider, editors, *ZB 2005*, volume 3455 of *Lecture Notes in Computer Science*, pages 375–393. Springer, 2005.
7. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000.
8. R. Canetti and M. Fischlin. Universally composable commitments. In J. Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 2001.
9. R. Cramer and I. Damgård. Multiparty computation, an introduction. Material for a course on Cryptologic Protocol Theory, Aarhus University, `www.daimi.au.dk/~ivan/CPT.html` (last checked April 17, 2007), 2004.
10. A. Datta, A. Derek, J.C. Mitchell, A. Ramanathan, and A. Scedrov. Games and the impossibility of realizable ideal functionality. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 360–379. Springer, 2006.
11. S. Delaune, S. Kremer, and M.D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
12. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.
13. A. McIver and C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, 2004.
14. Carroll Morgan. How to brew-up a refinement ordering. *ENTCS*, 2009. Proceedings of Refine 2009, to appear.
15. Carroll Morgan. The shadow knows: Refinement and security in sequential programs. *Sci. Comput. Program.*, 74(8):629–653, 2009.
16. M. Ying. Reasoning about probabilistic sequential programs in a probabilistic logic. *Acta Informatica*, 39(5):315–389, 2003.