Citation for published version

Boiten, Eerke Albert (2008) From ABZ to cryptography (abstract). In: Börger, Egon and Butler, Michael and Bowen, Jonathan P. and Boca, Paul, eds. Lecture Notes in Computer Science. Abstract State Machines, B and Z, First International Conference, ABZ 2008. LNCS, 5238. Springer-Verlag Berlin ISBN 978-3-540-87602-1.

DOI

https://doi.org/10.1007/978-3-540-87603-8₄0

Link to record in KAR

Document Version

Updated Version

From ABZ to Cryptography (extended abstract)

Eerke A. Boiten

Computing Laboratory, University of Kent, Canterbury, Kent, CT2 7NF, UK.
Email: E.A.Boiten@kent.ac.uk

Abstract. This paper reports on work in applying ideas from the ABZ world to modern cryptographic protocols. It describes the important differences between this and more "traditional" application areas, and a number of promising approaches in formal methods.

Disclaimer

The nature of this paper is such that a bibliography giving decent coverage of the problems raised and attempted solutions from both sides of the fence would take up more than the total space available here – the reader is invited to look elsewhere, e.g. papers and research proposals at [5].

1 Natural Bedfellows?

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 assortment.

2 Three Steps from the Ideal

Formal methods is about achieving correct systems. Ideally [12, 1], this correctness is achieved by construction: we use a "wide spectrum" language that encompasses both abstract specifications and executable programs, and transform one gradually into the other through small "correctness-preserving" steps. Refinement as a *process*, if you like, with the domain algebra, the properties of the problem, and a little creativity guiding us in creating a solution.

Slightly less desirable is post-hoc verification: proving that a proposed implementation is correct with respect to a specification (refinement as a *relation*), or that it satisfies certain properties. In the latter case, implementations and their properties may even be written in different languages.

If, given a specification and its intended solution, our mathematical framework does not help us in proving that it is correct, the next level is proof-checking. I.e., if someone comes along with a proof of correctness, we can formalise this, and then check mechanically that it discharges our overall proof obligation.

For modern cryptographic protocols (see below for what I mean by that), the state of the art is that proofs and proof methods are often insufficiently formalised for even proof checking to be a realistic prospective. So we are a full three steps away from the ideal way of achieving correctness.

3 Formal Methods and Cryptography

In the 1990s, formal methods techniques achieved major success in the modelling and analysis of cryptographic protocols, particularly work by the group using CSP around Oxford [9, 15] and by Paulson [13]. First, by considering non-deterministic choices of actions by the attacker, they allowed *abstraction* from attack *strategies* (and took anthropomorphism out of the equation: non-determinism encompasses “evil”). The second important aspect of this work was *automation*: using the theorem prover Isabelle in Paulson’s work, and using CASPER and the FDR[1] model checker in case of the Oxford group. However, this work was based on an abstraction of encryption which is an approximation. (Basically, the initial algebra assumption for encryption as the main constructor – implying an infinite algebra when all practical schemes work with fixed length bitstrings.) Thus, it may lead to false assurances of security. Also its emphasis on absolute notions of security does not sit well with modern cryptology.

4 Modern Cryptographic Protocols and Security

A modern cryptographic protocol may have the following properties:

- although its functionality is clear, its full set of desirable security properties may not be known yet;
- it contains explicit probabilistic elements, to mask input distributions and in “nonces”;
- its notion of security (correctness) is not an absolute one but approximate;
- moreover, this approximate correctness is relative to the computational resources available for an attack against it (which tends to imply an implicit probabilistic aspect);
- its security is not proved in an absolute sense but relative to the hardness of some computational problem;
- it uses primitives in a way which does not guarantee compositionality of the primitives’ properties.

All this means that the standard techniques and good intentions of formal methods do not work straight out of the box.

Many approaches to bridging the gap between formal methods and modern cryptography exist – see for example [4, 14, 7, 11, 8, 3]. These all have their advantages and disadvantages – but none are too close in spirit to the ABZ world.

5 What Do We Need, and What Has Been Done

Finally, I take a “bottom-up” view of how the ABZ world might approach the problem of “refinement for cryptographic protocols”: in which dimensions we would need to extend (say) standard Z states-and-operations refinement. This includes the following:

approximation Notions of correctness which are not exact but “close enough” – approximate refinement [6] would need to be strengthened to include fast convergence (“negligibility”). The cryptographic primitive of commitment, for example, requires two security properties – achieving both simultaneously is impossible, but schemes exist which approximate both with only negligible error.

probability Possibly protocols, and certainly attack models have a probabilistic element (“guessing”) to them. The work by McIver and Morgan [10] is a massive step forward in this area, and work on probabilistic refinement is continuing in several groups. Mingsheng Ying [16] has considered approximate probabilistic refinement.

action refinement Typical cryptographic protocols achieve a *single* objective through *multiple* communications between the parties involved. Thus, the granularity of actions decreases going from specification to implementation, requiring some kind of action refinement. Recent work by Banach and Schellhorn [2] is beginning to clarify issues of stutttering and upward vs. downward simulation in this area.

[1] The FDR tool is ©Formal Systems (Europe) Ltd.

attacks Protocols do not operate in isolation: multiple instances may run concurrently between different parties, and "dishonest" participants may not stick to the protocol. In the CSP work described above, this was modelled using non-deterministic choice over messages on a broadcast channel – is there an abstract data type analogue for this, and how do we model the limited ("polynomial") computing resources of such dishonest parties?

partwise and compositionality Refinement is monotonic with respect to most of the specification operators we use, allowing us to apply decomposition and partwise refinement. Approximation puts this under threat, and intuitively sensible notions of compositionality (e.g. [7]) have been shown to be unachievable for important cryptographic primitives.

All of this makes up a large research agenda to chip away at. Watch this space for a planned new EPSRC Network and new research in several of these areas.

References

1. R. Backhouse. *Program Construction: Calculating Implementations from Specifications.* Wiley, 2003.
2. R. Banach and G. Schellhorn. On the refinement of atomic actions. *ENTCS*, 201:3–30, 2008. Proceedings BCS-FACS Refinement Workshop 2007.
3. 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.
4. B. Blanchet and D. Pointcheval. Automated security proofs with sequences of games. In C. Dwork, editor, *CRYPTO'06*, volume 4117 of *Lecture Notes on Computer Science*, pages 537–554, Santa Barbara, CA, August 2006. Springer Verlag.
5. E.A. Boiten. Cryptography and formal methods project website. `www.cs.kent.ac.uk/~eab2/crypto/`
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. A. Datta, A. Derek, J.C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In *ICALP*, pages 16–29, 2005.
9. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *TACAS*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.
10. A. McIver and C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems.* Springer, 2004.
11. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In M. Naor, editor, *Theory of cryptography conference - Proceedings of TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151, Cambridge, MA, USA, February 2004. Springer.
12. C. C. Morgan. *Programming from Specifications.* International Series in Computer Science. Prentice Hall, 2nd edition, 1994.
13. L.C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.
14. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *ACM Conference on Computer and Communications Security*, pages 245–254, 2000.
15. P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and A.W. Roscoe. *Modelling and Analysis of Security Protocols.* Addison-Wesley, 2001.
16. M. Ying. Reasoning about probabilistic sequential programs in a probabilistic logic. *Acta Informatica*, 39(5):315–389, 2003.