

Commitment: A Challenge for Formal Methods

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. As an exploratory case study for the use of formal methods in cryptography, this paper considers the commitment primitive. Its reconstruction from an abstract informal characterisation explains most aspects of the usual commitment specifications, and demonstrates succinctly the inadequacy of the traditional formal methods “toolbox” in the context of modern cryptography. Additions to this toolbox are explored.

Keywords

Cryptography, formal methods, commitment, refinement.

1 Introduction

Commitment schemes form an essential ingredient of many cryptographic protocols. This paper introduces commitment as an exploratory case study for the use of formal methods in a modern cryptographic context. In order to do so, first an abstract characterisation is developed, and then a number of issues concerning the formal specification and verification of commitment schemes are explored. The particular value of using commitment as a case study is that it demonstrates that cryptography offers challenges which cannot be addressed by the convenient abstractions included in the “traditional” formal methods toolbox.

In the next section, we start by giving a reconstruction of commitment – trying to uncover all the assumptions and requirements that tend to be left implicit in descriptions in textbooks and research papers. Our first characterisation of commitment is informal and very abstract – as the introduction of a temporary imbalance in decision making between communicating parties. This already explains a number of decisions in actual commitment schemes, including the need for a commitment *function*, which is further elaborated in Section 3. Reasonable side conditions and the security requirements of commitment schemes point to the need for a computational approximate correctness relation. This is explored in Section 4, linking to previous work on approximate refinement. Following this, we briefly consider the “universal composability” abstraction, and the extent to which it applies to commitment. Finally, we summarise our contribution and list areas for further research.

2 Commitment as staggered causality

Commitment is a building block for cryptographic protocols, particularly in a context where the parties do not a priori trust each other. A *protocol* involves at least two parties and is, informally, an algorithmic prescription for a number of causally related communications between the involved parties, aiming to achieve a particular objective. Protocols being *prescriptive* rather than *descriptive* means that parties do not necessarily stick to the protocol. A protocol may succeed, or it may fail. It fails when the exchange of messages stops prematurely (though we are not considering time-outs explicitly here), for example after one party observes that another party is not adhering to the protocol. This could be evident from the format (“type”) of a received message, or something more subtle, such as a message not decrypting to an expected value. It succeeds if the protocol has completed (all its messages have been sent) without any of the parties declaring failure explicitly. If it succeeds, its objective must have been achieved. Parties which act according to the prescribed rules and aim to achieve the protocol objective are called “honest”. When at least one honest party is involved, the protocol succeeding *despite* the objective *not* having been achieved is a breach of security. *Correctness* of protocols is concerned with the avoidance of such breaches 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 failing, and in security analysis such actions may safely be disregarded¹.

Protocols may consist of fixed messages only, but in general they will be parameterised. The parameters which determine the contents of messages may include identities (addresses) of participants, pay load and other parameters associated with the protocol’s objective, and information generated by the execution of the protocol itself, such as timestamps and nonces.

Commitment schemes are associated with the latter kind of parameter, and their essence is a particular staggered notion of timing or causality. One party, the “committer”, takes the initiative by committing to a particular piece of information b , leading to a number of consequences and requirements for the parties’ future behaviour:

- The subsequent behaviour of the committer (i.e., its future messages) is *determined* by b , in the sense that it would be impossible for the committer to send the same messages after committing to $b' \neq b$; this is called the *binding* property.
- The subsequent behaviour of the other parties must be *independent* of the particular choice of b , in the sense that their behaviour cannot be distinguished from what it would have been in the case of $b' \neq b$; this is called the *hiding* property. In fact, this is required to hold for other parties irrespective of whether they stick to the protocol. This is essential, as commitments

¹ Clearly we are disregarding the kind of “denial of service” attacks where dishonest parties start up protocol runs but intentionally never complete them.

are typically used in contexts where there is no a priori trust between the parties.

- The hiding property may be lifted later by the committer revealing b in some way, this is called *opening* the commitment. From that point on, the other parties’ behaviour may also depend on b (and if they are honest, it usually *will* indeed depend). This is sometimes (e.g. [15]) called *correctness*, in the sense that the receiver needs to be able to open and correctly verify the committed value.

In the binding property we stated that the committer’s subsequent behaviour should be determined by b – we can and should be a bit more precise about that. Clearly that cannot be expected to apply to every subsequent message individually – the commitment scheme is likely to be part of a larger protocol, and some of its steps may not depend on the choice of b . As we are considering protocols that need not run to completion, we need to take a “trace semantics” (safety property based) view of behaviours: considering successful runs, as well as their prefixes. There is no point in insisting that a prefix of size zero is determined by b . However, we should be able to make the distinction for the next shortest prefix, i.e., a prefix of size one. As a consequence, a commitment “is” a communication. This makes sense: the committer taking a decision only becomes effective and binding once this fact is announced to “the world”! The first action being determined by b also ensures that by definition all traces following on from this will be determined by b .

One may wonder how this abstract definition prevents the committer from a “dishonest” opening of the commitment. This is ensured by the *other* parties’ subsequent behaviour depending on b . Commitments need not always be opened in a protocol, depending on the other parties – however, for any commitment made, there will be a possible continuation of the protocol in which it will need to be opened or where the committer will need to prove that the commitment was made. If the protocol gets to a point of opening the commitment, the value b will have to be revealed – if it is not revealed, it cannot be an opening. It is likely for the other parties’ side of the protocol to include a consistency check between the commitment message and the opening message, and to abort the protocol if this check fails. However, such a check is not directly enforced by this characterisation. Note also that a “commitment” which is immaterial for the future behaviour of all parties (e.g. one which is guaranteed never to be opened or proved consistent) is not a commitment, according to this definition.

The binding and hiding properties appear contradictory, particularly the fact that the commitment message should not be correlated to b for the other parties. In the next sections, we will show how this impacts on the notion of correctness to be used. In general, the main missing aspect of our informal description here is what it means for behaviours to be “distinguishable”.

Note 1. An alternative for the above characterisation could be given in terms of knowledge [19, 22], i.e., the committer “knows” b , and the other parties learn b at opening.

In the rest of this paper, we make three (common) restrictions on commitment schemes.

- we consider commitment to a single bit $b \in \{0, 1\}$. Multi-bit commitments can be built up by composing several instances of the bit commitment scheme, or in more efficient ways.
- we restrict ourselves to the case of a single other party, called the “receiver”. This is a true restriction: not all limitations of the two-party situation (see later) hold for larger numbers of parties [11].
- we only consider standard digital communication on noise-free channels, this excludes e.g. quantum commitment schemes [7].

3 The commitment message

Having established that committing *is* sending a message, we (for the moment) assume that the committer sends out a message

$$\mathit{commit}(b)$$

using some function $\mathit{commit} : \mathbb{B} \rightarrow \mathbb{Z}_k$. The result being an integer can be assumed without loss of generality, once we decide it has a bounded representation. It being *bounded* is a standard assumption for communication protocols of any kind, and quite reasonable from an efficiency viewpoint. In particular, it is required if the function commit is to be within some given complexity class, and if the total number of fixed-length protocol messages is to be bounded.

Moreover, we will assume that commit is a function whose definition is publicly available. This is not only in line with the normal situation in cryptography where e.g. encryption and decryption algorithms are assumed to be public (and only keys will be secret); it also serves two other (related) roles:

- the receiver receives c at commitment time, and (explicitly or implicitly) b if and when it is opened. In order to check that the committer is honest, the receiver will need to verify that $c = \mathit{commit}(b)$. The receiver cannot be allowed to know the inverse function of commit , as this would break the hiding property, so to perform this check he will need to know the function commit itself. The committer cannot be relied on as a trustworthy source for the function’s definition – we need to be able to trust the correctness of this definition unconditionally, and it being public guarantees that².
- commitments are a key element of Zero Knowledge protocols. Proofs of zero knowledge typically involve the construction of a simulation, showing that information available to the receiver (the “verifier”) suffices to generate transcripts of the protocol, which include commitments made by the “prover”. This is not generally possible if the function commit is secret. (See [7, property (iv)].)

² We will argue later on that it is unsatisfactory to found commitment schemes on established trust, as this is exactly what commitment is supposed to build up. The established trust in the commitment function is, essentially, what allows the “bootstrapping” here.

However, having established that the function *commit* is publicly available, we now run into a problem. Nothing stops a dishonest receiver, upon receiving the commitment c , from inverting *commit* by exhaustive search:

```
if  $c = \text{commit}(0)$  then  $b = 0$   
elseif  $c = \text{commit}(1)$  then  $b = 1$   
else fail (Invalid  $c$ , dishonest committer)  
endif
```

This clearly breaks the hiding property.

The normal solution for this is *randomisation*. The traditional argument for the need for randomisation in cryptography is being able to mask known distributions of plaintexts in encryption algorithms. The need for (hiding in) commitment schemes is an equally convincing argument.

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 [21]. 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. The former approach would not work in checking the commitment value at opening time, as there is no guarantee that two probabilistic calls $\text{commit}(b)$ would deliver the same outcome. Another way of looking at it is that we must ensure that exhaustive search of the domain of *commit* becomes less feasible, by artificially enlarging that domain. Thus, we are now looking at a commitment message of the form

$$\text{commit}(b, r)$$

where r is a number generated by the committer according to a particular probability distribution. For the zero knowledge simulation proof to work, once again this distribution would have to be publicly known. In the alternative view of probabilistic algorithms, this amounts to the probabilities for the algorithm’s internal choices being given – a reasonable assumption. The function *commit* as specified here is thus a “normal” deterministic function. The value r would need to be transferred to the receiver at opening time, so he can check that $\text{commit}(b, r) = c$.

The normal assumption is that r is bounded. There are several possible motivations for this, all quite reasonable:

- r needs to be communicated in the protocol, and thus needs to be bounded for efficiency reasons (e.g., for the overall protocol to have a fixed total number of bounded-size messages);
- the distribution of r needs to be uniform, or some other distribution which is only defined over a finite domain;
- the computation of the function *commit* needs to be efficient (e.g., within a given complexity class), and thus its inputs need to be bounded.

However, the property that r is bounded³ still does not exclude the possibility of exhaustive search of the domain of *commit*, which dishonest parties on either side can exploit:

The committer in order to invalidate the binding property can look for r' such that $\text{commit}(b, r) = \text{commit}(\neg b, r')$. If such an r' exists, we call $\text{commit}(b, r)$ ambiguous. Using r' , the committer could then open a different commitment to the one made originally. It follows that the function *commit*, to guarantee binding, should have the property that no such r' exists (for any r and b), i.e. there is no ambiguity. Thus we require:

$$\forall b, r \bullet \neg \exists r' \bullet \text{commit}(b, r) = \text{commit}(\neg b, r')$$

The receiver in order to invalidate the hiding property can look for r and b such that $c = \text{commit}(b, r)$. This will only be useful if the value of b found is the same for all possible such r , i.e., for all b', r' if $\text{commit}(b', r') = c$ then $b' = b$. Thus, to guarantee hiding in every case, the function *commit* should ensure that whenever $c = \text{commit}(b, r)$, there also exists r' such that $c = \text{commit}(\neg b, r')$, i.e., it should have maximal ambiguity. We require:

$$\forall b, r \bullet \exists r' \bullet \text{commit}(b, r) = \text{commit}(\neg b, r')$$

The consequence of all this is of course that no function *commit* satisfying both desirable properties exists, and its specification using the traditional formal methods toolbox is equivalent to **false**.

This makes commitment an even more difficult example to cover symbolically than encryption. The black-box Dolev-Yao assumption (pairing and encryption as the constructors of an initial algebra) only leads to an inconsistency if we explicitly state that the collection of terms (bitstrings of a given length) is finite. Although the reconstruction above included boundedness arguments explicitly, this actually was not necessary to exhibit the intrinsic inconsistency between hiding and binding. In other words, the idealised black box specification of commitment leads to an inconsistency *even* if we do *not* restrict ourselves to a finite set of messages.

However, commitment *is* considered useful, even if practical schemes cannot live up to the ideal. The compromise of 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, we can find schemes that approximate both as closely as required, provided we assume (dishonest) parties being constrained to bounded (polynomial) time.

The next section is an exploration of this kind of approximation, phrasing it in program correctness terminology in addition to the probabilistic and complexity-theoretic idiom that is more usual in the cryptographic context.

³ Actually, r being recursively enumerable, e.g. an unbounded integer, is already sufficient.

4 Computational approximate correctness

We first introduce the general ideas as they arose in a program correctness setting, and then apply them to hiding, to binding, and to the combination of the two. Finally, we make some observations on this particular way of characterising computational security.

4.1 Quantitative correctness specifications

The need for approximate notions of correctness has been identified previously in the formal methods community, albeit for different reasons: there is often a mismatch between abstract idealistic specifications and their implementations using bounded resources – to the extent that, strictly speaking, the constraints imposed by bounded resources invalidate correctness. Several solutions have been proposed for this, including permissive forms of correctness such as “retrenchment” [2]. In joint work with John Derrick, the author has proposed approximate refinement [4, 5] which uses converging chains of specifications as approximations of ideal solutions, characterised as follows.⁴

Definition 1 (Quantitative Correctness Specification) A quantitative correctness specification (QCS) in language \mathcal{L} consists of:

- a countably infinite *sequence of specifications* S , i.e., $S : \mathbb{N}^+ \rightarrow \mathcal{L}$, and
- a *metric* d on⁵ \mathcal{L} , i.e., $d : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}^+$.

such that S has a limit s according to d , i.e.,

$$\forall \epsilon > 0 \bullet \exists n : \mathbb{N}^+ \bullet \forall m : \mathbb{N}^+ \bullet m > n \Rightarrow d(S(m), s) < \epsilon$$

A sequence S is considered equivalent to its limit. A QCS is called *simple* if all its elements are equivalent, i.e., $d(S(n), s) = 0$ for all n . \square

As the distance function is defined on semantic equivalence classes, limits are only unique up to semantic equivalence. In particular, if the distance function captures only one aspect of correctness, the limit represents all specifications which display “ideal” behaviour in that respect. Metrics defined in [4, 5] use, for example, the number of system operations to highlight the difference between specifications (based on [16]).

Slowly coming back to the cryptography area, one metric that could be used is statistical distance⁶ between the probability distributions generated by indexed

⁴ This definition is simplified from the one in [5] which uses indexing by any ordered set without maximal elements. In particular, using any infinite subset of \mathbb{N} is also possible.

⁵ Actually, on the semantic equivalence classes of \mathcal{L} – equivalent specifications have zero distance.

⁶ See Appendix A for definitions of statistical distance, negligibility, and a number of other basic notions.

families of probabilistic algorithms. Consider, for example, a desirable distribution D , and an indexed family of probabilistic algorithms A_x which approximate D . These would be called *statistically indistinguishable* if their statistical distance were negligible in the size of x . This would imply A_x being a quantitative correctness specification under statistical distance as the metric, with D as its limit. However, it being a QCS is a strictly weaker property, as QCSs do not capture the “fast” convergence to the limit implied by the distance being *negligible* rather than just tending to 0.

Thus, we strengthen the concept accordingly:

Definition 2 (Strong Quantitative Correctness Specification) A strong quantitative correctness specification in language \mathcal{L} is a quantitative correctness specification (S, d) with a limit L such that $d(S(x), L)$ is a negligible function in x .⁷ \square

4.2 Hiding

The hiding aspect of commitment schemes can now be expressed as a strong QCS using statistical distance to compute the distance function. However, there is a twist: we cannot use the statistical distance between perfect hiding and the effect of *commit*, but rather use the distance between committing to 0 or to 1. (It is clear that the main relevance of the distribution generated by *commit* is the extent to which it allows one to distinguish between commitments to 0 and 1 rather than the actual values generated; in particular, for every perfectly hiding scheme, isomorphic ones at statistical distance almost 1 also exist.)

Let us be more precise about the type of the function *commit* now: its extra argument r and its result will be taken as bitstrings rather than numbers (to avoid having to compute “length” of numbers); moreover, the function will be a family, indexed by the length of its randomising argument (the so-called “security parameter”), which will also determine the length of the result. So, we have

$$\text{commit}_n : \mathbb{B} \times \mathbb{B}^n \rightarrow \mathbb{B}^{k(n)}$$

Now, its statistical hiding effect *sh* is defined as⁸:

$$\text{sh}(\text{commit}_n) == SD(\text{commit}_n(0, r), \text{commit}_n(1, r')) \text{ for } r, r' \in_u \mathbb{B}^n$$

where \in_u denotes uniform probabilistic choice from a set. In general, for characterising the approximate correctness of cryptographic schemes, this is the point

⁷ The proof that this indeed characterises L as a limit is simple: for any ϵ , instantiate the polynomial $p(x)$ to the constant $1/\epsilon$.

⁸ The instances of commit_n on the right hand side are really distribution transformers, transforming uniformly distributed random variables r and r' into distributions over (part of) the range of *commit*. Properly writing them as such would do away with the variables r and r' and the anomaly of having to introduce two names for the same uniform distribution in order to remove suggestion of dependence. See Dan Grundy’s thesis [18] for further considerations on the notations etc. for the theories that underlie cryptography.

at which the particular “attack game” [3, 24] that characterises its security needs to be inserted – the function sh is the “adversary’s advantage”, which should be 0 in order to achieve perfect security. This is, of course, already the required distance to the limit; in order to compare all possible functions we set

$$sd(f, g) == | sh(f) - sh(g) |$$

where the limit L has $sh(L) = 0$ by definition. A commitment scheme which has perfect hiding as its limit according to this characterisation can be said to be “statistically hiding”, i.e., hiding is not perfect but breaking it is independent of the computational power of a dishonest receiver.

The relation between the standard formulation of security and the above characterisation is represented by the following.

Theorem 1. $(\{n : N \bullet commit_n\}, sd)$ is a strong QCS iff *commit* is statistically hiding. \square

However, the more interesting notion of approximate correctness is *computational* hiding. Rather than taking the raw distributions generated by committing to 0 or 1, this includes a “distinguisher” D , a polynomial probabilistic algorithm⁹ which operates on the values generated and returns a boolean representing its “guess” of the value committed to. Applying this distinguisher thus turns a distribution over the co-domain of *commit* into a distribution over booleans.

Unfortunately, the introduction of computational complexity takes the definitions outside the “constructive” formal methods area. We need a universal quantification over all such distinguishers D . Due to the lack of an induction principle for probabilistic polynomial algorithms, to prove such a quantified predicate we need to proceed by contradiction and reduction: all D satisfy predicate P if the existence of a D satisfying $\neg P$ leads to a contradiction. The contradictions that are normally aimed at are based on the common conjectures of complexity theory: that no algorithm in a given complexity class exists that solves a particular “hard” problem. The proof is then by reduction: a hypothetical algorithm D satisfying $\neg P$ is used as a black box subroutine (“oracle”) in an algorithm (of the correct complexity) solving the problem. Thus, as one might have expected, the introduction of a computational notion of correctness brings in complexity theory as a necessary foundation. Correctness, then, is conditional on the assumed computational difficulty of a collection of “pedigree” problems such as factoring and discrete logarithms. This dependence on “likely” but unproven assumptions is a property of the research area as a whole, and for that reason we will not worry about this here.

The hiding advantage of using a particular distinguisher D is defined as

$$ch_D(commit_n) == SD(D(commit_n(0, r)), D(commit_n(1, r'))) \\ \text{for } r, r' \in_u \mathbb{B}^n$$

⁹ Damgård and Nielsen [14] note that in the general definition of computational security this may be a non-uniform algorithm, but that this makes no difference in this context – indeed all numbers generated by *commit_n* have the same length.

This represents the total probability that D will return a different output for values from the distributions $commit_n(0, r)$ and $commit_n(1, r')$. It is not necessary for D to return the exact value b when it discovers (a high probability that) the commitment was to b ; just as long as it returns different outputs for the different distributions “as often as possible”. The hiding advantage is then defined as that of the “best” possible distinguisher from all probabilistic polynomial algorithms $PP(n)$, and the distance is defined as before:

$$\begin{aligned} ch(commit_n) &== \mathbf{max}\{D : PP(n) \bullet ch_D(commit_n)\} \\ chd(f, g) &== |ch(f) - ch(g)| \end{aligned}$$

The relation between our characterisation and the standard one [14] is embedded in the following.

Theorem 2. $(\{n : N \bullet commit_n\}, chd)$ is a strong QCS iff *commit* is computationally hiding. \square

4.3 Binding

Having characterised hiding through QCS, we need to do the same for binding. The “attack game” in this context is to find ambiguous values that can be opened as commitments to either boolean value. As we have presented things so far, with a fixed function *commit* to be used in all runs of the commitment scheme, this attack can be done “off-line”: given the function *commit*, a dishonest committer only needs to find a *single ambiguous value*, which it then can use in every protocol run. (Compare this to the dishonest receiver’s only off-line attack against the hiding property: to build up an inversion table for *all* of the co-domain of *commit*, which is computationally infeasible if *commit* is computationally hiding.) For that reason, commitment schemes which are not perfectly binding will normally have an additional parameter to prevent pre-computation of ambiguous values by dishonest committers. This makes off-line attacks much less effective (though still possible). Of course, in any run of the scheme, the additional parameter’s value should be provided by the receiver¹⁰.

With this particular attack game in mind, we define the distance to the limit for any function f of the type of *commit* as the attacker’s advantage. The attacker A is a probabilistic algorithm polynomial in n which returns a pair of numbers.

$$\begin{aligned} cb_A(commit_n) &== Prob(commit_n(0, r) = commit_n(1, r')) \\ &\quad \text{where the output of } A \text{ is } (r, r') \\ cb(commit_n) &== \mathbf{max}\{A : PP(n) \bullet cb_A(commit_n)\} \\ cbd(f, g) &== |cb(f) - cb(g)| \end{aligned}$$

¹⁰ In schemes where commitment is based on encryption, the additional parameter is typically a key. The presence of such a key is simply assumed in [14]. This reconstruction only partially explains half of it (namely, for perfect hiding). In general, it needs to be set up from the side that stands no chance of cheating anyway – why it is needed as well, set up by the committer, for perfect binding is not yet explained here. The off-line attack may not be the whole story.

Now we can characterise the standard security notions in terms of QCSs again.

Theorem 3. $(\{n : \mathbf{N} \bullet \text{commit}_n\}, \text{cbd})$ is a simple QCS iff *commit* is perfectly binding. \square

Theorem 4. $(\{n : \mathbf{N} \bullet \text{commit}_n\}, \text{cbd})$ is a strong QCS iff *commit* is computationally binding. \square

4.4 Hiding and Binding

In order to combine hiding and binding, we define a one-dimensional distance from the two distance measures, for example by

$$cd(f, g) = \sqrt{\text{cbd}(f, g)^2 + \text{chd}(f, g)^2}$$

The pair $(\{n : \mathbf{N} \bullet \text{commit}_n\}, cd)$ appears to be a meaningful expression of a computationally secure commitment scheme (provided cd gets small enough quickly enough for large enough n), but it being a QCS according to the above definition depends on whether its limit exists. I.e., do we allow “infeasible” specifications such as perfect binding *and* hiding to exist (and be distinguished) in our specification language? The preceding discussion suggests we might not; and the identification of a sequence of feasible specifications with an infeasible limit in QCSs may still be problematic. However, in the next section, we observe that a specification of a perfect commitment scheme can be given in a way that does not appear infeasible at all, by relaxing its context.

4.5 Conclusions

This section presented a reformulation of existing notions of security in a program correctness framework. This serves a conceptual purpose, and will allow some cross-fertilisation between the two areas. Conceptually, it strengthens the view that security is “just” a form of correctness: on the one hand a liberal one which only requires properties of interest to hold “often enough”; on the other hand one where the properties of interest will change between the different protocols and schemes and may not always be obvious in advance. In the practical sense, it means that methods for the verification of approximate correctness can be used to verify security properties. Further research into approximate correctness is planned, using the application in cryptography as a motivation and source of examples.

5 Third parties and universal composability

State-based formal methods (such as Z, VDM, ASM) have well-developed theories of correctness and refinement. The approach to cryptography verification that is conceptually closest to this area is that of universal composability

(Canetti [9]), also known as reactive simulability (Backes, Pfitzmann and Waidner [23, 1]). These methods allow abstract specifications of cryptographic protocols in terms of secure interactions with a trusted intermediary (the so-called “ideal model” [17]). Security of a particular protocol is then proved by showing that any interaction (in any environment) by an adversary with the protocol (in the “real model”) can be *simulated* in the ideal model, normally allowing for an observational difference that is negligible in the security parameter. As a consequence, the analysis of more complicated protocols invoking the proved protocol can use the “ideal model” abstraction instead of the actual protocol. This is what is called “universal composability” (UC). On the face of it, this is a very promising approach, as it allows for compositional reasoning, even if the elementary proofs are normally still by reduction.

The specification of (perfect) commitment in the UC model is very simple. To commit to a bit b , the committer sends b securely to the intermediary. The intermediary then signals to the receiver that the committer has committed. To open the commitment, the committer sends an opening request to the intermediary, who will then pass b on to the receiver. The hiding property is guaranteed because the communication from committer to intermediary is secure and the intermediary is trusted. Binding is also guaranteed because the intermediary is honest. This, together with the security of communication between intermediary and receiver, also guarantees correct opening.

We have now shown that the ideal model specification ensures the three properties of our abstract specification – however, not the reverse: that the ideal model poses no additional constraints. Interestingly, researchers do not agree on which kind of specification should come first. Proponents of UC argue that there is always a risk of important security properties being overlooked, whereas the ideal model will ensure all relevant properties as well as properties preventing unthought-of attacks. (See, e.g., the discussion on specifying e-voting in [12].) Conversely, e.g. Datta et al [15] argue that the properties (embedded in game conditions) should prevail, and that there may be several “ideal models” to realise them.

This ideal model specification itself does not come close to being an acceptable implementation, for several reasons. The first is a principled one: commitment is a building block for establishing trust, so to implement it by assuming a trusted host is somewhat circular. In any case, if a trusted host is available there is limited need for commitment. Second, our abstract characterisation does not allow for a trusted host in this role: it is neither subject to hiding (it knows b), nor to binding (it is in a position to change b), so it is neither a committing nor a receiving party.

A more substantial problem with commitment in this context is that UC commitment schemes cannot exist without further assumptions. This is proved in detail in [10, 15] – there exist environments and adversaries which require the ideal model simulator to know the value committed to, even before the commitment is opened. Universally composable commitment schemes with additional set-up assumptions are given in [10] (providing computational hiding and bind-

ing) and [13] (allowing perfect hiding or binding). In both of these, both parties are assumed to have access to a common reference string with a prescribed distribution. Universal composability is clearly a desirable property, but maybe in some contexts its security properties make it an unrealistically strong abstraction. Datta et al [15] discuss alternatives by taking games as the starting point.

6 Concluding comments

The contributions of this paper are the following:

- a new characterisation of commitment at a very high level of abstraction;
- a reconstruction of a cryptographic primitive that convincingly but succinctly shows why traditional formal methods techniques do not suffice;
- a connection between novel ideas in formal methods and central notions of security.

Each of these allows for further exploration. The abstract characterisation could be formalised in epistemic logics [19, 20] and it will help in linking commitment to related abstractions such as opacity [8] and ignorance-preserving refinement [22]. The obvious next step from this reconstruction is to look at existing commitment protocols and their proofs – the abstractions and proof techniques used in those, and how these reflect back on the ideas in this paper. An exploration of the link between “refinement” and (complexity-theoretic) reduction is given in [6]. Further work on approximate refinement is ongoing.

Acknowledgements

I would like to thank Dan Grundy for first raising my interest in cryptography and many discussions on this subject. A number of visits, discussions and seminars have been very helpful and supportive: Nigel Smart (who suggested I attend MCP 2006 and pointed out salient properties of commitment); Mark Ryan and his group; Steve Schneider; Peter Ryan and Robert Stroud at Newcastle; and Stefan Kahrs and others of the Kent TCS group. Ivan Damgård and Jesper Buus Nielsen wrote the excellent introduction [14], and answered several email questions.

References

1. M. Backes, B. Pfitzmann, and M. Waidner. A general composition theorem for secure reactive systems. In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2004.
2. R. Banach and M. Poppleton. Retrenchment, refinement and simulation. In J.P. Bowen, S. King, S. Dunne, and A. Galloway, editors, *ZB 2000*, volume 1878 of *Lecture Notes in Computer Science*, pages 304–323. Springer-Verlag, 2000.
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. 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.
5. E.A. Boiten and J. Derrick. A quantitative approach to program correctness. Submitted for publication, 2008.
6. E.A. Boiten and D.C. Grundy. Reduction and refinement. *ENTCS*, 201:31–44, 2008. Proceedings REFINÉ 2007: the 11th BCS-FACS Refinement Workshop.
7. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.
8. J. Bryans, M. Koutny, L. Mazaré, and P.Y.A. Ryan. Opacity generalised to transition systems. In T. Dimitrakos, F. Martinelli, P.Y.A. Ryan, and S.A. Schneider, editors, *Formal Aspects in Security and Trust*, volume 3866 of *Lecture Notes in Computer Science*, pages 81–95. Springer, 2005.
9. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000.
10. 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.
11. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19. ACM, 1988.
12. 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.
13. I. Damgård and J.B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer, 2002.
14. I. Damgård and J.B. Nielsen. Commitment schemes and zero-knowledge protocols. Material for a course on Cryptologic Protocol Theory, Aarhus University, www.daimi.au.dk/~ivan/CPT.html (last checked July 1, 2008), 2008.
15. 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.
16. J. W. de Bakker and J.-J. C. Meyer. Metric semantics for concurrency. In J. W. de Bakker and J. J. M. Rutten, editors, *Ten Years of Concurrency Semantics: Selected Papers of the Amsterdam Concurrency Group*, pages 104–130. World Scientific, Singapore, 1992.
17. 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.
18. D. Grundy. *Concepts and Calculation in Cryptography*. PhD thesis, Computing Laboratory, University of Kent, 2008.
19. J.Y. Halpern, R. van der Meyden, and M.Y. Vardi. Complete axiomatizations for reasoning about knowledge and time. *SIAM J. Comput.*, 33(3):674–703, 2004.
20. S. Kramer. Logical concepts in cryptography. Cryptology ePrint Archive, Report 2006/262, 2006.
21. A. McIver and C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, 2004.

22. C. Morgan. *The Shadow Knows: refinement of ignorance in sequential programs*. In T. Uustalu, editor, *Mathematics of Program Construction, 8th International Conference*, volume 4014 of *Lecture Notes in Computer Science*, pages 359–378. Springer, 2006.
23. 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.
24. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004.

A Definitions

This section contains a number of definitions that are well-known in theoretical cryptography, listed here for completeness.

Definition 3 A function f with argument x is *negligible* in $n(x)$ if it decreases faster than the reciprocal of any positive polynomial p in $n(x)$, i.e.,

$$\forall p \bullet \exists N \bullet \forall x \mid n(x) > N \bullet f(x) < \frac{1}{p(n(x))}$$

□

This definition is usually stated with n instantiated to the identity function and then generalised implicitly. Another option is for $n(x)$ to be the size (base 2 logarithm) of x ; this is clearly weaker, e.g., $1/x$ is already negligible in the size of x . If $n(x)$ does not go to infinity when x does, it makes little sense to say that f is negligible in $n(x)$.

Discrete probability distributions on a domain are given as functions from that domain to the probabilities assigned to each value.

Definition 4 Given two probability distributions P and Q on a common domain Y , their *statistical distance* is defined by

$$SD(P, Q) == \Sigma_{y \in Y} \mid P(y) - Q(y) \mid$$

□