

# Cryptographic Commitment and Simultaneous Exchange

Peter Bardsley  
The University of Melbourne

Andrew Clausen  
The University of Melbourne and the University of Pennsylvania

Vanessa Teague  
The University of Melbourne

## **Remark 1** *Version 1.6.9*

Mutually beneficial exchange is a fundamental building block in the conceptual framework through which economists view the world. Yet simultaneous exchange is not entirely unproblematic. Unless the exchange is truly simultaneous, there is a risk that the person who first relinquishes control of their asset will be left with nothing if the other person does not reciprocate. The institution of exchange requires a host of supporting institutions in order for us to be confident that it will work.

If I buy a ticket at a local railway station, seller and buyer are separated by a transparent screen, pierced by a rotating platform. The seller puts the ticket on one side, the buyer puts the money on the other, and the platform rotates to implement the exchange. If I buy a loaf of bread at the corner store, then there is an informal but effective set of social conventions that support the expectation that I will not walk out without paying for the bread, and that the shopkeeper will not put the money in the till without providing the goods. If one buys a house, then the transaction generally involves a stage where the parties, or their representatives, sit around a table signing, sequentially witnessing and exchanging documents in a carefully orchestrated manner. Custom, reputation, repeated interaction, the involvement of trusted intermediaries, and the existence of a legal system that recognises and enforces legal rights and obligations are some of the institutions that support exchange.

Exchange at a distance is more problematic than face to face exchange. This is especially so if communication is asynchronous, and simultaneity of actions cannot be guaranteed. The sophisticated communication capability created by the Internet has greatly increased the opportunities for mutually beneficial exchange. Buyers and sellers can find one another, and mutually advantageous opportunities can be discovered very easily. However exchange between strangers at a distance can be hazardous, especially if they are in separate legal jurisdictions. There may be problems in verifying that the goods are as described, in verifying that the agents are who they claim to be, in enforcing the agreement, and so on. We shall not be concerned with such problems, in which synchronicity is not the essential issue, but will focus on the fundamental underlying problem of simultaneous commitment. We shall inquire how two agents, fully informed about the terms of trade, can agree to a mutually beneficial exchange, and simultaneously commit to this exchange, through the interchange of messages in an asynchronous environment.

Contracts do not provide an immediate solution to the simultaneous exchange problem. Even if a court could enforce any exchange contract, the problem reappears when the traders sign

the contract. Since a half-signed contract may be selectively enforced by the remaining party, exchanging signatures at arms length is as hazardous as exchanging other items.

It is clear that we might want to use a trusted intermediary, either to carry out the transaction, or to monitor it and punish any infraction. We could both of us give our object to the intermediary, knowing that if the other party does not do so then our object would be returned. But the conditions that would sustain our trust in such an intermediary are not straight forward. We consider the simple unmediated exchange between two individuals to be the more fundamental transaction, and we will focus on this problem.

Since we will be concerned with communication games in which players send messages to one another, in our environment ownership of an asset will amount to knowledge of a secret. For example, my secret might be the number of a Swiss bank account; or it might be the key that unlocks an encrypted document that contains an enforceable, legally binding contract signed with my digital signature; or of a document that transfers the legal title to an asset to another party; or it might be a digital coin. Assets are exchanged if the secrets that embody them become known to the other party. The ability to implement exchange is intimately connected with the attainment of simultaneity of commitment. Without this simultaneity, one agent can walk away with both objects, so the possibility of exchange collapses. The attainment of this simultaneity in an asynchronous environment appears to be quite a delicate matter.

The structure of the paper is as follows. In Section 1 we introduce the Blum-Damgard gradual exchange protocol, which has been put forward by cryptographers as a solution to the simultaneous exchange problem, and we also discuss the literature. In Section 2 we outline the environment in which we will work. In Section 3 we set out, in a reasonably informal manner, the main cryptographic constructs and concepts that will be used. Our intention is to explain these ideas to an audience to whom they may not necessarily be familiar. In Section 4 we set up the exchange mechanism and analyse its properties, deferring to Section 5 details of the cryptographic implementation of our protocol. Conclusions are summarised in Section 6, and technical details of the cryptographic proofs are set out in the Appendix, which we hope is reasonably self contained.

## 1 Commitment through Gradual Exchange

Cryptographers have already paid attention to the simultaneous exchange of secrets. Blum (1983a) made the fundamental suggestion that if the secrets could be made divisible then pieces of the secret could be exchanged alternately one by one. Divisibility can be achieved by encrypting each secret with a long digital key. If each key consists of, say, 100 digits, then these digits can be exchanged one by one. The intuition is that the hold-up problem implicit in asynchronous exchange of information might be solved in a manner similar to the staged gradual mutual investment approach to the investment hold up problem (Pitchford & Snyder 2004). It is interesting to note that biologists have recently discovered evidence of similar mechanisms in the reproductive strategies of certain organisms<sup>1</sup>. Damgard (1995) presented a more robust protocol to implement Blum's proposal. Damgard writes<sup>2</sup> of the exchange problem:

If the two secrets are represented as bit strings of the same length, this can be solved by exchanging the secrets bit by bit; if this is done honestly, no party will be more than one bit ahead of the other; put another way: if at some point  $A$  can compute  $B$ 's secret in time  $T$ , then  $B$  can compute  $A$ 's secret in at most time  $2T$  by guessing the bit he may be missing.

---

<sup>1</sup>See (NewScientist 2005) for an account of sperm trading between hermaphroditic sea slugs.

<sup>2</sup>The quotation is verbatim, except that  $s_A$  and  $s_B$  were replaced with  $A$ 's secret and  $B$ 's secret, respectively.

There are two issues that arise in considering such a proposal. The first is the credibility of the partial information that is being exchanged. Is it possible for one agent to convince the other that the bits that are transferred really are part of the hidden keys, as asserted, and to do so safely, without accidentally revealing more information than was intended? Blum and Damgard resolved this credibility issue through the use of the powerful cryptographic technique of probabilistic "zero knowledge proofs" (Goldreich, Micali & Wigderson 1991, Goldreich 2001). The second is whether the divisibility of the keys really solves the exchange problem when rational agents are acting strategically.

In this paper we examine the Blum-Damgard rational exchange claim formally, inquiring as to whether their protocol can actually be implemented as the equilibrium of a game played by rational agents. When posed in this form, the answer is clearly no: cryptography relies on the fact that some calculations can be designed to be computationally intractable. Although possible in theory, to carry out such a calculation is prohibitively expensive. However a fully rational agent can see through any logical chain of deduction, no matter how complex. In particular they can carry out any finite computation<sup>3</sup>; they can crack any encryption scheme based on computational complexity. Cryptography does not work for such hyper-rational agents. Cryptography relies on a weak but precise assumption of bounded rationality, and on a conjecture about the hardness of certain algorithms (for precise statements see Section 3). We are interested in the extent to which this minimal degree of bounded rationality enlarges the set of economic institutions that we can design. The question that we ask is whether simultaneous exchange can be supported as the equilibrium of a message exchange game of the Blum-Damgard type played between such boundedly rational agents.

We find that the Blum-Damgard intuition does not entirely hold up. Despite the divisibility introduced by the bit-by-bit gradual exchange protocol, there remains an intrinsic indivisibility: there is a moment when one of the players becomes irrevocably committed to the exchange, and this is crucial to understanding the equilibrium. We find that the Blum-Damgard protocol fails for a range of parameters. Despite this, through a refinement of their approach (essentially through a finer control of computational cost) we are able to support simultaneous exchange under quite weak assumptions.

Our work is related to the computer science literature on exchange protocols (Blum 1983a, Damgard 1995, Jakobsson 1995, Goldreich 2004). It is related to the economics literature on mediated and unmediated communication games (Forges 1990, Gerardi 2004, Forges & Koessler 2005), and to the issues of simultaneous communication, credible communication, coordination and jointly controlled lotteries that arise with asynchronous cheap talk; see Aumann & Hart (2003), BenPorath (2003), Forges & Koessler (2006); see also Teague (Forthcoming). It is also related to the literature on incremental investment and hold up (Pitchford & Snyder 2004).

Forges (1990) shows that the equilibrium payoffs available in mediated and unmediated communication games coincide if there are four or more players. BenPorath (2003) and Gerardi (2004) consider outcomes that can be achieved when the minimum number of players is between three and six. More recently, Forges & Koessler (2005) study mediated communication with partially verifiable information, but leave the relationship with unmediated communication an open question. Credibility of communication is important but unmodelled in these papers. In contrast, we have only two players in an unmediated environment (but we do impose extra structure on the problem) and we require credibility of all messages, which we establish through cryptographic techniques (Goldreich et al. 1991). Simultaneity is quite significant in cheap talk models (Aumann & Hart 2003). While it is reasonably straightforward for players to enforce simultaneous choice of messages with "cryptographic commitments", simultaneous transmission

---

<sup>3</sup>As emphasised by Frege and Russel, any properly formulated mathematical statement can be reduced to an assertion in pure logic.

requires a secret exchange of the type that we model. So our results potentially have applications to cheap talk models in general.

The computer science literature on unmediated cryptographic communication is reviewed by Goldreich (2004) in Chapter 7. Generally, in this literature agents are modelled as either “honest” (meaning that they follow the assigned protocol perfectly) or “malicious” (meaning that they may perform any computationally feasible actions), and the analysis considers whether the malicious agents can cause the honest ones to produce the wrong output or reveal secret information. Recently computer scientists have also become interested in protocols for rational agents, rather than malicious or honest ones, producing a literature drawing on both game theory and computer science (Dodis & Rabin 2007, Halpern 2007). The exchange problem is addressed in the computer science literature by Syverson (1998), and further considered by Buttyan & Hubaux (2001) and Buttyan, Hubaux & Capkun (2002). Unlike these authors, we do not rely on an externally imposed punishment for cheating, as Syverson’s protocol does, and we require our mechanism to be subgame perfect. Subgame perfection is essential to credible implementation in our sequential environment. Syverson and Buttyan also assume that communications can be designed to be fully credible, an assumption that goes beyond what can be achieved through zero knowledge proofs. Sandholm’s analysis of unenforced e-commerce transactions (Sandholm 1996, Sandholm 1997) does implement a subgame perfect equilibrium. However, he considers only intrinsically divisible goods that can be divided up directly, and the issues of the divisibility of information and the credibility of partial information exchange do not arise.

In this paper we formalise an asynchronous exchange environment, and characterise the credible secret exchange mechanisms that implement exchange as a subgame perfect equilibrium in this environment. We then show that, if trade is individually rational, then sufficient credibility to support exchange can always be established by cryptographic cheap talk.

As with any mechanism, an issue arises of commitment of the agents to the mechanism itself. In particular, in any incremental exchange mechanism there is always a temptation to hold up the last transfer and to renegotiate the rules of the game that is being played. The incomplete contracts literature has addressed this issue in the context of incremental investment and exchange; see for example Pitchford & Snyder (2004). We have nothing to add here, and in common with the rest of the literature we assume that agents play the game as designed. Our protocols will allow exchange increments to be as small as desired.

## 2 The Environment

Before embarking on an analysis of the exchange problem we will outline the environment that we have in mind. We describe an idealised environment, based loosely on institutions in the market for real property, that will be sufficient to illustrate the exchange problem. It is useful to formalise the environment in order to clarify the assumptions that we make. The concepts that we focus on are assets (these are the valuable things that agents may wish to exchange), ownership of assets, and transfer of ownership. We also need to specify the communication environment. In our environment exchange of assets will be equivalent to exchange of certain secrets.

Let us consider for a moment the market for real property (for example houses, cars, shares). An asset can be conceptualised as a bundle of property rights — a set of permissible actions that an agent (the owner of the property rights) can take, including the right to prevent actions by other agents. These rights are typically embodied in a physical legal document, a title deed, that specifies these rights. If an agent can prove that he or she is the owner of an asset, then they may exercise the rights embodied in the title deed. In the background, there is a legal system (the

court) that gives value to these assets by enforcing the rights that they embody. The functions required of the legal system are

1. To certify the scope and validity of an asset; that is to certify that it embodies a bundle of rights that the court will enforce, and to determine whether an action or set of actions is consistent with these property rights. For example, this might mean determining whether a document is genuine or a forgery, or determining whether rights over a particular parcel of land fall within the scope of a title deed.
2. To link assets to owners. For example an agent might prove to the court that they can produce a signature (a physical, ink signature on a piece of paper) that links them to the asset. This link might be via a chain of valid transfers of ownership going back to some original well accepted owner, or it might be by verifying that the signature matches that in a central database (an asset register) of certified owners.
3. To enable transfer of ownership of an asset from one agent to another.

To abstract from this we introduce a simple cryptographic concept of identity. There are many approaches to establishing and authenticating identities in communication systems. For a comprehensive review, see Goldreich (2004), who points out the fundamental role of *proof of knowledge* in almost all identity authentication systems. That is, one proves one's identity by proving that one knows some secret that nobody else can be expected to know. This applies even to traditional ink signatures, in the sense that I know how to produce my written signature, and it is very unlikely that any else knows how to do so (unless they are a skilled forger). Here we will make the simple assumption that the identity of an agent is specified by an encrypted secret known only to the agent. Agents can prove their identity to another party (the court, or to a potential purchaser of the asset) by proving knowledge of the hidden secret. We will refer to the secret as the password associated with the identity. We will discuss what this means in more detail below, but we should point out that authenticating one's identity will mean proof that with very high probability one knows the secret. Henceforth we will define an identity for an agent to be such an encrypted secret (so an agent may have more than one identity).

To formalise the environment we assume

1. There is one (or more<sup>4</sup>) asset registers. An asset register is a database of assets (title deeds linked to identities). A title deed is just a digital document registered in an asset register. An identity is an encrypted secret. The asset register is public, and its entries can be examined and copied by anybody.
2. An agent owns an asset if they own the identity associated with an asset; that is, if they know the secret password associated with the identity.
3. Each asset register provides a safe and reliable method by which one agent may prove to any other agent that they are the owner of an asset. That is, if and only if they are the owner, they can prove that they know the secret password associated with the asset without actually revealing the password. Details of how this can be implemented using standard cryptographic techniques will be discussed below, in Section 3.

---

<sup>4</sup>We allow the possibility that there may be several asset registers both for realism (assets may be in different jurisdictions) and because we do not want the asset register to act in some way as a trusted intermediary facilitating exchange.

4. Each asset register provides a safe and reliable method by which the owner of an asset can change the identity associated with an asset. That is, they can generate a new identity, with a password known only to them, unlink the asset from the old identity and link it to the new identity.
5. To facilitate transfers, an agent with identity  $I_1$  who owns an asset can lock the database for a brief fixed time period in favour of another agent with identity  $I_2$ ; if the database is locked then the only transaction that is permitted is to transfer ownership to the other agent by unlinking the asset from  $I_1$  and linking it to  $I_2$ ; this transaction can only be initiated by an agent who knows the passwords for *both*  $I_1$  and  $I_2$ . If the time limit expires without a transfer occurring, the password is automatically updated to a fresh one that was generated beforehand (so the same agent still owns the asset, but has a new identity associated with it).
6. It is common knowledge that the password to any new identity generated by an asset register will be an independently drawn random uniformly distributed integer between 1 and some large fixed integer  $p$ , and that it is known only by the owner of the identity.

The last assumption is for convenience when we model the exchange game. It ensures that it is common knowledge that the secrets are independently uniformly randomly distributed when the exchange game starts. Details of how this can be implemented will be discussed below, in Section 3; it requires that the asset register has access to a random number generator such as a thermal noise device, and is trusted to cooperate appropriately in generating passwords<sup>5</sup>.

Exchange of assets can now be implemented purely by passing messages that exchange secrets. The first agent locks her asset in favour of the second agent (who has generated a new identity, with a password known only to himself, for the purpose of this transaction). In a similar transaction the second agent locks his asset in favour of the first. They then exchange secrets: their original passwords (not the new passwords that they have just generated). They each now have the authority to unlink the other agent from their newly acquired asset, and the exchange is complete.

We note that the role of the legal system is minimal. It plays no part in the exchange and exists in the background only to give value to the assets. We also note that, unlike what we assume about assets, we do not need the identity of agents to be authenticated by a trusted third party. Nor do we need identities to be permanently or transparently associated with agents.

Given this framework, we will consider protocols or mechanisms whereby agents may exchange messages through a secure but asynchronous communication system. We have in mind a secure email link where delivery of messages within a fixed finite time is reliable, but subject to a random delay with some known upper bound. It is thus very difficult to achieve simultaneous action in such a system. We assume that there is no legal framework governing the messages that may be sent.

### 3 The Cryptographic Infrastructure

In this section we will outline in a relatively simple way some of the standard cryptographic concepts that we will use. We will focus on one way functions and zero knowledge proofs, which are the essential building blocks that we will use later, and indicate how they may be used in the implementation of an asset register as discussed in the previous section. More technical

---

<sup>5</sup>We could manage without this assumption at the cost of added modelling complexity, relying on the agents generating their secrets in the most unpredictable manner possible as part of the equilibrium.

cryptographic details will be deferred to Section 5 and the Appendix. In this section we will make some simplifying assumptions<sup>6</sup> that will be relaxed in the more technical sections.

Modern cryptography works through careful manipulation of computational cost (Goldreich 2001). For computational cost to be relevant in a game theoretic context we need two assumptions. Note that the first of these is an explicit assumption of bounded rationality.

**Assumption 1 (bounded computational ability)** *Agents incur a cost<sup>7</sup> proportional to the number of computational steps required for a computation.*

**Assumption 2 (No side calculations)** *Agents do not make any side calculations that are not specified as moves in the game. In particular, they don't attempt to guess a secret unless guessing is explicitly part of a strategy. When updating beliefs they do not engage in additional search, using only search results that are already to hand.*

This assumption requires that agents play the game as specified, and do not step outside the game by taking unmodelled additional (computational) actions. We make this assumption as we want all decisions about costs to be explicitly associated with strategies, and not hidden in side calculations. Otherwise it would be possible for agents to carry out expensive calculation, searching for their opponent's secret while refining their beliefs, without these costs being included in payoffs.

**Assumption 3 (existence of computationally intractable problems)** *There exist problems, parametrised by a security parameter  $k$ , whose computational cost increases in  $k$  at an asymptotic rate faster than any polynomial in  $k$ .*

For a precise statement of the computational complexity assumptions required to support cryptography see Goldreich (2001); these assumptions are related to, but not the same as, the well known  $\mathcal{P} \neq \mathcal{NP}$  conjecture. We will approach the issue of computational complexity essentially from an engineering point of view: by choosing an appropriate value for the security parameter  $k$  it is possible to design computational tasks for which it is common knowledge that the cost of undertaking them far exceeds any conceivable benefit in the environment under discussion. We will refer to such tasks as infeasible. For example, encryptions can be designed so that it is infeasible to break them, and we will generally assume this of any encryptions unless the contrary is specifically noted. More generally, we will require strategies to be computationally feasible in the sense that they cannot be conditional on quantities that cannot feasibly be computed.

Using computational complexity it is thus possible to design computational tasks whose expected cost is essentially infinite (computationally infeasible) or essentially zero (computationally trivial). For our purposes this degree of control will be too coarse. We will want to exercise much finer control over computational cost. This is a more delicate matter, to which Section 5 is devoted.

### 3.1 One way functions

The most basic cryptographic technique involves the use of *one way functions*. A function is one way if it can be computed efficiently but computing a preimage is infeasible. For example, given a large prime  $p$ , let  $\mathbb{U} = \{1, 2, \dots, p-1\}$ . Then it is known that  $\mathbb{U}$  is a cyclic group under

<sup>6</sup>In particular, it is useful for technical reasons to work not in the full group of units  $\mathbb{U}$ , as discussed here, but in a cyclic subgroup of high prime order, and to use a more complicated cryptographic commitment method. Details are in the Appendix.

<sup>7</sup>We regard this cost as a given characteristic of the computational technology. If there is a market for computational services then this would be the price in this market.

multiplication mod  $p$ . This means that there is an element  $g \in \mathbb{U}$  such that the powers of  $g$  generate  $\mathbb{U}$ : every element  $x \in \mathbb{U}$  is of the form  $x = g^n \bmod p$  for some unique  $n \in \mathbb{U}$ . The function  $f : \mathbb{U} \rightarrow \mathbb{U}$  defined by  $f(n) = g^n \bmod p$  (the *discrete exponential* to base  $g$ ) is believed to be a one way function. The inverse, the *discrete logarithm*, is well defined (it can in principle be calculated by an exhaustive search over all the elements of  $\mathbb{U}$ ) but it is infeasible to compute if  $p$  is large enough. We will choose a large, fixed prime  $p$  that will be kept fixed throughout this section. If  $p$  is chosen properly, it is easy to find a generator  $g$  (Menezes, van Oorschot & Vanstone 1997). We will assume that such a generator has been fixed and is common knowledge.

We will use the notation  $[x] = g^x$  for applying the discrete exponential to  $x$ . The notation  $[x]$  is chosen to suggest that  $x$  has been placed in a locked box or sealed envelope. For example, if  $p = 11$  and  $g = 7$  then  $[8] = 9$  since  $7^8 \bmod 11 = 5764801 \bmod 11 = 9$ . The important properties of  $[x]$  are as follows.

- given  $x$ , it is very easy to calculate  $X = [x]$ ; however if  $p$  is large, given  $X$  it is very difficult to find the  $x$  such that  $X = [x]$ . In principle one can search for  $x$  over the whole search space  $\mathbb{U}$  but this is very expensive. See Koblitz (1994) for the computational intractability of the discrete logarithm problem.
- if  $[x] = [y]$  then  $x = y \bmod p$
- $[x + y] = [x][y]$ ; this is just the law of exponents
- if Alice<sup>8</sup> knows the  $x$  such that  $[x] = X$  then she can prove (probabilistically) to Bob that she knows  $x$ , and she can do so without revealing any other information about  $x$ ; we will discuss below how this can be done

If Alice has a secret, encoded<sup>9</sup> as a number  $s \in \mathbb{U}$ , and she sends  $S = [s]$  to Bob, then she is said to make a *cryptographic commitment* to  $s$ . Under our assumptions of computational intractability she has not revealed  $s$ , but she cannot later claim that her secret was  $t \neq s$  since Bob can compute  $T = [t]$  and compare it with  $S$ . It is now clear how to implement identities in the environment described in Section 2. An identity is just an encrypted secret. If  $s$  is a secret password known to Alice, then the identity that will link her through this password to an asset in the database is just the discrete exponential  $S = [s]$ .

With some help from the asset register, Alice can guarantee that her secret is uniformly distributed in the range  $[1, p-1]$ , and commonly known to be so. Alice and the authority generate her secret jointly. First, she generates<sup>10</sup> some  $s'$ , independently and uniformly distributed in the required range and sends the authority  $[s']$ . Then the authority<sup>11</sup> generates a “blinding factor”  $b$  (also supposed to be independently and uniformly distributed in the same range) and sends it to Alice. Alice’s secret would then be defined as  $s = s' + b \bmod p - 1$ , which the authority could generate directly from  $[s']$  and  $b$ . The secret is correctly distributed if either Alice or the authority generated their contribution correctly and keep it secret; for it to be common knowledge it is necessary that the authority be trusted to generate the blinding factor correctly. There are many variations on this idea, including more authorities and different trust models. See (Goldreich 2004) for more details.

<sup>8</sup>Traditionally, cryptographic games are played between Alice and Bob, and sometimes Carol and David as well.

<sup>9</sup>Any text string can be encoded as a single, rather large, integer using Godel numbering or some variant thereof (Davis, Weyuker & Sigal 1994)

<sup>10</sup>We assume that the agents, and the asset register, have access to independent random number generators, perhaps using some hardware such as a thermal noise device.

<sup>11</sup>or some trusted third part source of randomness

In order to exercise her property rights, Alice must be able to prove her identity to another party. That is, she must be able to prove that she knows the key  $s$  associated with the identity  $S = [s]$  without revealing  $s$ . We now show how that can be done.

### 3.2 Zero knowledge proofs

Alice's cryptographic commitment  $S$  becomes more interesting if she can credibly reveal to Bob some properties of  $s$  without actually revealing  $s$ , or indeed any leakage of other information about  $s$ . The cryptographers have developed a powerful theory of probabilistic *zero knowledge proofs* (ZKP) which codifies this idea. In particular, Alice may wish to prove that she knows  $s$  without revealing anything else (a *zero knowledge proof of knowledge*). In general, a zero-knowledge proof is a game between Alice, who asserts that a proposition  $P$  is true, and Bob who needs to be convinced. The structure of the game is that Bob issues challenges to Alice by asking a series of tricky questions. Alice wins the game if she can always answer Bob's questions satisfactorily; otherwise Bob wins. She may guess a satisfactory answer a few times, but if she is faking and  $P$  is actually false then she will eventually be found out. The proof should satisfy three conditions: it should be *complete*, meaning that Alice can always win if  $P$  is indeed true; it should be *sound*, meaning that Alice must eventually fail with arbitrarily high probability if  $P$  is false, and finally it must be *zero knowledge*, meaning that no information is revealed to Bob other than the fact that  $P$  is true. See (Goldreich 2001, Goldreich 2002) for details, in particular for a formal definition of what it means that there is no leakage of information beyond the proposition that is being proved. If Alice convinces Bob of a proposition  $P$  by executing a probabilistic proof with the property that a false assertion will be detected with probability at least  $1 - \varepsilon$ , then we will say that her assertion is  $\varepsilon$ -credible.

We will illustrate the concept by describing how Alice can prove to Bob that (with very high probability) she knows an  $s$  such that  $[s] = S$ , and can do so without revealing  $s$  or releasing any other information about  $s$ . This is exactly what she needs to do to prove her ownership of an asset in the environment of Section 2.

Alice and Bob both know  $S$ . Alice claims that she knows  $s$  such that  $[s] = S$ . The ZKP proof is an interactive game, played in multiple rounds as follows.

- Protocol 1 (Proof of knowledge of discrete log)**
1. Alice chooses a uniformly distributed random number  $x \in \mathbb{U}$  and reveals the commitment  $X = [x]$  to Bob.
  2. Bob tosses a coin.
  3. If Bob's coin comes up heads, then he challenges Alice to reveal a number  $x$  such that  $[x] = X$ . If she does so, and Bob confirms that  $[x] = X$ , then the round ends; if Alice cannot meet the challenge then she must have cheated and Bob wins the game. This branch of the protocol guarantees that she does not cheat and release a fake commitment  $X \neq [x]$ .
  4. If Bob's coin comes up tails, he challenges Alice to reveal  $s + x$ ; that is to say, a number  $z$  such that  $[z] = SX$ . If she does so, and Bob confirms that  $[z] = SX$ , then the round ends; otherwise Bob wins the game. This branch of the protocol guarantees that she knows  $s$ , since if she has not released a fake commitment then she knows an  $x$  such that  $[x] = X$  and she has just shown that she knows a  $z$  such that  $[z] = SX$ . But  $[z] = SX = [s][x] = [s + x]$  so  $z = s + x$ . Thus she must know  $z$ .

If Alice does in fact know the  $s$  such that  $[s] = S$ , then it is clear that she can always meet Bob's challenge by acting exactly as required in the protocol. If not, she may try to fake, and guess whether Bob will challenge her to produce  $x$  or  $s + x$ . In the first case she expects to be

asked to produce  $x$  and she can follow the protocol, choosing  $x$  randomly and revealing  $X = [x]$ . In the second she expects to be asked to produce  $x + s$ . She can choose a number  $z$  randomly and reveal  $X = [z]/S$  (this is clearly a fake commitment — she does not actually know an  $x$  such that  $[x] = X$ ). When challenged she will reveal  $z$ . However she can bluff in this way only if she can anticipate which challenge Bob will issue; eventually she will be caught out. If enough rounds are played then the probability that she is just bluffing can be made as low as desired. At each stage Bob learns only a random number  $x$ , or a random number  $s + x$  (but not both!). So what he sees is just a series of independently drawn random numbers. He can generate such a random series for himself, so he cannot learn anything from the transcript of the game apart from the fact that Alice seems always to be able to win.

## 4 The Exchange Game

Before getting into formalities, it may be useful to have an intuitive idea of the message passing game that will be played. Imagine that Alice breaks up her secret into pieces from which it can be reconstituted. She writes each of these pieces on a slip of paper, seals it in an envelope (which Bob cannot open), and passes the sealed envelopes to Bob. During the game she opens some of the envelopes for Bob. The message that has been passed at any stage is the contents of the envelopes that have been opened (the length of the message will be the number of envelopes that have been opened). Once some envelopes have been opened, Bob has partial information about Alice’s secret. This may not allow him to reconstitute her secret, but it may provide him with information that helps if he tries to guess (reducing his computation cost). As information is passed back and forth, the guessing costs move along a zigzag exchange path on a grid in computation cost space (see Figure 1). The information that Bob can extract from Alice’s messages, and the cost grid that the exchange moves along, depend on the rules—the decomposition protocol that has been used.

Alice and Bob have secrets  $\alpha$  and  $\beta$ , which we will also refer to as assets. The values of these assets, to Alice and Bob respectively, we will write as  $(\alpha_A, \alpha_B)$  and  $(\beta_A, \beta_B)$ . We assume that Alice’s trade surplus,  $\beta_A - \alpha_A$  and Bob’s trade surplus,  $\alpha_B > \beta_B$  are positive, so that they both wish to trade. We assume that the secrets have been randomly encoded as numbers  $s_A \in \mathbb{S}$  and  $s_B \in \mathbb{S}$  in some suitable set  $\mathbb{S}$ . Without loss of generality we may assume that the secrets are these numbers, and write  $\alpha$  or  $s_A$  indiscriminately (and do the same with respect to  $\beta$  and  $s_B$ ). We will discuss the exchange game from the point of view of Alice (from Bob’s point of view the situation is symmetric), and in the interest of readability we will write  $s, s^i, S, S^i, \sigma$  rather than  $s_A, s_A^i, S_A, S_A^i, \sigma_A$  in the following paragraphs.

First, Alice breaks up her secret  $s$  into a list  $\sigma = (s^1, \dots, s^N) \in \mathbb{S}^N$  of  $N$  pieces from which it can be reconstituted (for simplicity we will assume that  $N$ , the number of pieces into which the secret is broken up, is the same for both Alice and Bob). The exact way that the secret is decomposed will be specified in the protocol; we do not wish at this stage to be more specific. Let  $\Omega = \mathbb{S} \times \mathbb{S}^N$  be the set of all possible secrets  $s$  and decompositions  $\sigma$ , and let  $\Pi \subset \Omega$  be the set of secrets and decompositions that are consistent with the protocol.

**Example 1 (Blum Damgard)** *Blum and Damgard specify a protocol in which the  $s^i$  are the digits in a binary expansion of  $s$ ; in that case we would have*

$$\Pi = \left\{ (z_0, (z_1, \dots, z_N)) \in \Omega : z_0 = \sum_i z_i 2^i, z_i \in \{0, 1\} \ (i \neq 0) \right\}.$$

Alice encrypts her secret  $s$  and the pieces  $s^i \in \sigma$ . She sends Bob the encrypted versions  $S$  and  $S^i$ . Following standard terminology, we will refer to these as (cryptographic) *commitments* to the encrypted values. The first commitment  $S = [s]$ , which contains Alice’s secret, can be

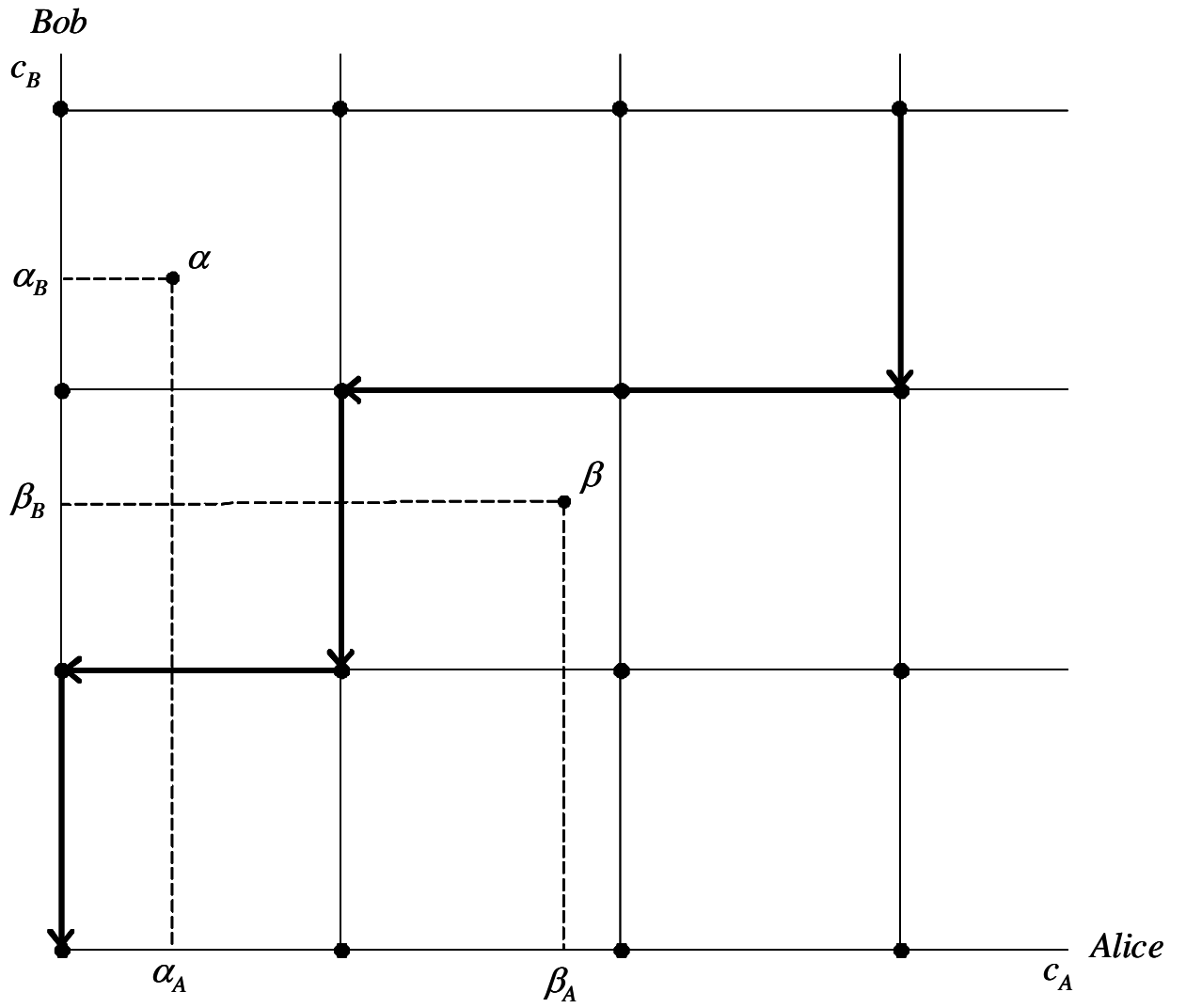


Figure 1: The exchange path must lie in the grid  $G$ .



opened if one knows or can guess  $s$ . Bob can thus attempt to open  $S$  by guessing values for  $s$  and trying them out, though this may take rather a long time. The pieces  $s^i \in \sigma$  are encrypted in a way that leaks no information to Bob, even if he has infinite computing power available.<sup>12</sup> The only reason Alice gives him these pieces is to prevent her from lying later on. Thus Bob can open the  $S^i$  only with Alice's help. We assume that it is infeasible for Alice to open commitments falsely, that is to be able to open one envelope in two different ways. (By infeasible, we mean that the expected cost greatly exceeds any possible gains in the subsequent exchange game.) Alice then asserts that  $(s, \sigma) = (s, (s^1, \dots, s^N)) \in \Pi$ , that is that the secret  $s$  has been decomposed properly and can be reconstituted from the pieces  $s^i$ . Since Bob knows only encrypted versions of  $s$  and the  $s^i$ , it is infeasible for him to evaluate the validity of this assertion without further help from Alice.

It will be convenient below to refer to the commitment  $S^i$  by its index  $i$ . We think of  $S^i$  as a locked envelope labelled with the number  $i$ , containing  $s^i$ . It is impossible to derive  $s^i$  from  $S^i$ , even with infinite computational resources. We write  $E = \{1, \dots, N\}$  for the set of sealed envelopes (commitments) that Alice has passed to Bob, containing pieces of her secret (in our notation we identify each envelope with the number with which it is labelled).

Alice and Bob now alternately exchange messages, opening some of their commitments and revealing information about their secrets. A message  $\tau$  from Alice will consist of a set of envelopes  $E_\tau \subset \{1, \dots, N\}$  that are to be opened, and a function  $\tau : E_\tau \rightarrow \mathbb{S}$  that specifies the contents of each of these envelopes;  $\tau(i)$  is the value that lies inside envelope  $i$ . This communication will be supported by sending Bob keys that will enable him to verify the veracity of the message<sup>13</sup>. We write  $|\tau|$  for the length of the message, that is number of envelopes that are opened in message  $\tau$ ; it is just the number of elements in  $E_\tau$ . We write  $\mathcal{M}$  for Alice's message space. That is,  $\mathcal{M}$  is the set of partial functions  $\tau : E \rightarrow \mathbb{S}$ . Messages, being partial functions, are ordered by set inclusion. If  $\tau \subset \tau'$  then we will say that  $\tau'$  extends the message  $\tau$ . We note that the original decomposition  $\sigma = (s^1, \dots, s^N)$  of Alice's secret is such a message; sending this message would amount to opening all of the envelopes in  $E$ .

To understand the meaning of these messages it is useful to establish some notation. Recall that  $\Pi \subset \Omega = \mathbb{S} \times \mathbb{S}^N$  is the set of allowable decompositions. We write  $\pi : \Pi \rightarrow \mathbb{S}$  for the projection onto the first factor. That is,  $\pi(s, \sigma) = s$ . We make the following definitions:

1. If  $\tau$  is a message then  $\Omega_\tau = \{(s, \sigma) \in \Omega : \sigma \subset \tau\}$ ; it is the set of all pairs  $(s, \sigma)$  that are consistent with the message  $\tau$
2. If  $\tau$  is a message then  $\Pi_\tau = \Pi \cap \Omega_\tau = \{(s, \sigma) \in \Pi : \sigma \subset \tau\}$ ; it is the set of all  $\Pi$  decompositions  $(s, \sigma)$  that are consistent with the message  $\tau$
3. If  $\tau$  is a message then  $\mathbb{S}_\tau = \{s \in \mathbb{S} : \exists \sigma : (s, \sigma) \in \Pi_\tau\}$ ; it is the set of all secrets that are consistent with the message  $\tau$ ;  $\mathbb{S}_\tau$  is the image of  $\Pi_\tau$  under the projection  $\pi$

If  $\Pi$  is not clear from the context then we will write  $\mathbb{S}_{\Pi, \tau}$  rather than  $\mathbb{S}_\tau$ . Having received a message  $\tau$ , Bob knows (if he is sure that Alice has been truthful) that  $s \in \mathbb{S}_\tau$ , and if he tried to guess her secret then this would be his search space. If he believes her secret to be uniformly distributed<sup>14</sup>, conditional on  $\tau$ , then his expected search cost is proportional<sup>15</sup> to  $|\mathbb{S}_\tau| - 1$ . By a

<sup>12</sup>using a Pederson commitment: see the Appendix for details

<sup>13</sup>We can assume that Alice sends only messages that open successfully with the keys that she provides; any messages that do not open can simply be deleted and ignored.

<sup>14</sup>A sufficient condition to justify such uniform updating of Bob's beliefs is that he believes that Alice randomises amongst her messages. By this we mean that she chooses her message  $\tau$  randomly amongst the possible messages  $\tau'$  permitted by the protocol with  $\mathbb{S}_\tau = \mathbb{S}_{\tau'}$ . We will henceforth assume that Bob believes that Alice always randomises in this way, and ensure that any equilibrium strategy that we construct has this property.

<sup>15</sup>If  $|\mathbb{S}_\tau| = 1$  then the set is a singleton, and no searching is necessary.

slight abuse of notation we use to the size of Bob’s search set as a measure of his expected search cost:

**Definition 1 (Search Cost)**  $C(\tau) = |\mathbb{S}_\tau| - 1$

In general, the cost  $C(\tau)$  depends on the content of the message  $\tau$  (*which* commitments have been opened) not just on the size  $|\tau|$  of the message (*how many* have been opened). However in many cases some or all of the pieces of Alice’s secret are symmetric: it does not matter which of these commitments have been opened, only how many, and there is a function  $c$  (depending on  $\Pi$ ) such that for these commitments  $C(\tau) = c(|\tau|)$ . That is, if the number of messages that have been sent is a sufficient statistic for the computation cost then we can write the computation cost as a function  $c(|\tau|)$  of the number of messages sent. In general, given such a function, we will make the following definition:

**Definition 2** *A message  $\tau$  is permissible if  $C(\tau) = c(|\tau|)$ .*

In the Blum Damgard protocol, when Alice tells Bob one of her digits, she eliminates half of the possible remaining numbers, halving the size of the search space. Clearly this protocol satisfies the assumption that the expected search cost depends only on the number of digits that have been revealed, not on which ones have been revealed, and all messages are permissible. The Blum Damgard grid is exponentially spaced, and the expected cost function is  $c(p) = 2^{N-p} - 1$ , where  $p$  is the number of pieces of Alice’s secret that have been revealed. We later argue that this is inadequate for most exchanges, and develop a protocol with a grid that is linear and as finely grained as desired.

**Example 2** *In the protocol that we will develop below, Alice will have a special envelope  $i^*$  which must be opened last. A message  $\tau$  will be permissible if it opens  $i^*$  last: that is, if either  $i^* \notin E_\tau$  or  $E_\tau = E$ . If  $\tau$  is permissible then  $c(|\tau|)$  will be linear in  $N - |\tau|$ .*

As Alice and Bob exchange pieces of their secrets, their computation costs move between points on the grid  $G = G^A \times G^B = C(\mathcal{M}) \times C(\mathcal{M})$ . If they restrict themselves to permissible messages then their computation costs move between points on the grid  $G_0 = G_0^A \times G_0^B = c(\mathbb{N}) \times c(\mathbb{N})$  (see Figure 1). For simplicity, we will assume that the values  $\alpha_A, \alpha_B, \beta_A, \beta_B$  do not lie on the grid  $G$ . This means we can ignore cumbersome knife-edge cases.

It is clear that the credibility of Alice’s assertions is crucial. Bob knows only encrypted versions  $S, S^i$  of her secret  $s$  and its components  $s^i$ . Alice asserts that  $(s, (s^1, \dots, s^N)) \in \Pi$ , and hence that Bob’s expected search costs evolve according to the function  $C(\tau)$ . If her assertion is false, then  $s$  might have no relation to the alleged components  $s^i$  that are being progressively revealed, and Bob will be cheated in the exchange. Blum and Damgard supported their exchange by a zero knowledge proof, which reassures Bob that Alice is acting honestly, and does so without allowing any of her valuable information to leak out. We will do the same. In the protocol set up below the agents can ensure by conducting probabilistic proofs that their assertions are  $\varepsilon$ -credible, where  $\varepsilon > 0$  is some very small probability fixed in advance of the game. It is useful however first to consider the exchange game in the case where  $\varepsilon = 0$ , when all true assertions can be proved with full credibility, and then to consider the more realistic case  $\varepsilon > 0$ . We defer detailed specification of the cryptographic protocol to Section 5.

## 4.1 Exchange with fully credible proofs

The exchange begins as above with Alice and Bob decomposing their secrets and exchanging encrypted versions. In the full game (with  $\varepsilon > 0$ ) they would conduct a series of probabilistic

proofs about how the secrets had been decomposed, in order to establish credibility. Here we will simply assume that the secrets have been decomposed honestly and that they have put forward fully credible proofs of this. In this section we will characterise the exchange paths that can (under this honest decomposition assumption) be supported by a subgame perfect equilibrium of the exchange game.

**Assumption 4** *The decomposition protocol is fully credible*

**Corollary 1 (Monotonicity)** *The expected search cost  $C(\tau)$  is monotone in  $\tau$ . If Alice reveals more information then Bob's search set  $\mathbb{S}_\tau$  (weakly) decreases and cannot grow bigger.*

**Proof.** Monotonicity is a consequence of Bayesian updating and the credibility of Alice's assertions. ■

We will also make the following assumption, which will be in force for the rest of the paper, about the decomposition protocol:

**Assumption 5** *The protocol  $\Pi$  is such that for all secrets  $s$  the decomposition  $\sigma$  is unique. That is, the projection  $\pi : \Pi \rightarrow \mathbb{S}$  is bijective.*

**Corollary 2** *For any message  $\tau$  the expected cost of searching for  $s$  given  $\tau$  is, to within a normalisation constant,  $C(\tau) = |\mathbb{S}_\tau| - 1$*

**Proof.** Since the projection  $\pi : \Pi \rightarrow \mathbb{S}$  is a bijection, and Bob knows that Alice's secret  $s$  is uniformly distributed in  $\mathbb{S}$ , he knows that the decomposition  $(s, \sigma)$  is uniformly distributed in  $\Pi$ . Since her assertions are fully credible, he believes by Bayesian updating conditional on the message  $\tau$  that the decomposition is uniformly distributed in  $\Pi_\tau$  and hence that  $s$  is uniformly distributed in  $\mathbb{S}_\tau$ , the projection of  $\Pi_\tau$  (recall the assumption above that Alice randomises amongst her possible messages). ■

The following assumption, which will be relaxed later in this section, simplifies the exposition.

**Assumption 6 (Permissibility)** *All messages are permissible.*

This assumption means that the commitments (envelopes) are all symmetrical or interchangeable, in the sense that they each contain the same quantum of information, and that they can be opened in any order. As a consequence of this assumption agents need only choose how many commitments to open, not which ones. It also follows that expected search cost, conditional on a message  $\tau$ , depends only on the length of a message (the number of commitments opened), not on its content:  $C(\tau) = c(|\tau|)$ . Thus the cost grid  $G = C(\mathcal{M}) \times C(\mathcal{M})$  is the same as the permissible grid  $G_0 = c(\mathbb{N}) \times c(\mathbb{N})$ .

After exchanging encrypted versions of their secrets and encodings, Alice and Bob engage in a conversation, taking turns in sending messages to each other. A conversation

$$\begin{aligned} h &= (n_A, n_B) \\ &= ((n_A^1 \leq n_A^2 \leq n_A^3 \leq \dots \leq n_A^k), (n_B^1 \leq n_B^2 \leq n_B^3 \leq \dots \leq n_B^l)) \end{aligned}$$

will be a pair of increasing sequences  $n_A$  and  $n_B$  of non-negative integers. Here  $n_A^i$  is the number of commitments that Alice has opened after she has spoken  $i$  times. We require that the agents provide additional information each time they speak, or the conversation terminates. So we require that  $n_A^1 \neq 0$  (unless  $k = 0$ ) and that all the inclusions  $n_A^{i-1} \leq n_A^i$  be strict except possibly for the last one. If  $k = 1$  and  $n_A^k = 0$ , or if  $n_A^{k-1} = n_A^k$  then we will say that message  $k$  is *terminal*: it is Alice's turn to speak but she has declined to reveal any more information. Such a message

terminates the whole conversation. We will write  $\bar{n}_A = n_A^k$  for the total number of commitments that Alice opens in the conversation. We make corresponding definitions for  $n_B$ . A conversation thus defines an increasing zigzag path  $((n_A^0, n_B^0), (n_A^1, n_B^0), (n_A^1, n_B^1), (n_A^2, n_B^1), \dots, (\bar{n}_A, \bar{n}_B))$  in  $\mathbb{N} \times \mathbb{N}$ . Here we write  $(n_A^0, n_B^0) = (0, 0)$  for the initial state before anyone speaks.

A computational cost sequence

$$\begin{aligned} c &= (c_A, c_B) \\ &= ((c_A^1, c_A^2, c_A^3, \dots, c_A^k), (c_B^1, c_B^2, c_B^3, \dots, c_B^l)) \end{aligned}$$

is a pair of sequences  $c_A$  and  $c_B$  of numbers defining a zigzag path  $((c_A^0, c_B^0), (c_A^1, c_B^0), (c_A^1, c_B^1), (c_A^2, c_B^1), \dots, (\bar{c}_A, \bar{c}_B))$  that lies in the computational cost grid  $G$ . Here we write  $(c_A^0, c_B^0) = (c^0, c^0)$  for the initial cost before anyone speaks and  $(\bar{c}_A, \bar{c}_B) = (c_A^k, c_B^l)$  for the terminal costs. We say that the cost sequence is monotone if  $c_A^{i-1} \geq c_A^i$  and  $c_B^{i-1} \geq c_B^i$  for all  $i$ , and that it is *complete* if  $(\bar{c}_A, \bar{c}_B) = (0, 0)$ .

A conversation  $h$  thus defines a monotone computational cost sequence  $c = c(h)$  where  $c_A^i = c(n_B^i)$  is the expected computational cost that Alice would face if she tried to guess Bob's secret after he had spoken  $i$  times and revealed  $n_B^i$  pieces of his secret, and  $\bar{c}_A = c(n_B)$ . Note the reversal of indices in the notation:  $c_A^i$  is Alice's expected cost after Bob has spoken  $i$  times. A conversation thus defines a decreasing zigzag path in the cost grid  $G$ . When the conversation terminates, as it must eventually do, the agents play a search game  $\Gamma(h)$  where they may, if they wish, search for each other's secrets incurring expected search costs  $(\bar{c}_A, \bar{c}_B)$ .

To summarise, the state of the exchange game is described by a conversation history  $h$ , which implies a monotone cost path  $c(h)$  in the cost grid  $G$ . A history is terminal if the last player chose not to extend the conversation, declining to open any more commitments. The strategy space is determined by specifying the actions that can be taken at each state. At a non-terminal history  $h$  the active player, let us say it is Alice, must choose an integer  $m_A = m_A(h)$ , the (cumulative) number of commitments she will open, such that  $\bar{n}_A \leq m_A \leq N$ . If she chooses  $m_A = \bar{n}_A$  then she is choosing to terminate the conversation; if she chooses  $m_A > \bar{n}_A$  then she must open  $m_A - \bar{n}_A$  additional commitments. At a terminal history  $h$  Alice and Bob simultaneously choose search strategies in the search game  $\Gamma(h)$ . Alice's search strategy  $\gamma_A(h)$  lists a sequence of numbers to inspect in  $\mathcal{S}$ . Alice proceeds through the sequence until she either finds Bob's secret or she reaches the end of the sequence. Since their decisions are strategically independent, and they learn nothing useful in the course of the search that would change their strategies, they will make one of two choices in the search game. They will either not search, or they will search the entire set of possible secrets consistent with the messages that they have received, stopping only if they find what they are looking for. Since Alice believes any possible secret is equally likely, she finds any search order is optimal. We focus on equilibria in which Alice chooses a random search order.

We now study which cost sequences can be supported in a subgame perfect equilibrium.

**Lemma 1** *In the guessing game  $\Gamma(h)$  Alice will attempt guess Bob's secret if and only<sup>16</sup> if  $\bar{c}_A < \beta_A$  and Bob will attempt to guess Alice's secret if and only if  $\bar{c}_B < \alpha_B$ .*

Since the only incentive the players have to give each other information is the promise of more information in the future, no trade is an equilibrium.

**Lemma 2** *Silence is an equilibrium. That is, the message policies  $m_A(h) = \bar{n}_A$  and  $m_B(h) = \bar{n}_B$ , combined with the guessing strategies above form a subgame perfect equilibrium.*

<sup>16</sup>By the assumption that none of the valuations lie on a grid point, we do not need to consider cases of equality.

**Proof.** By the one-deviation principle, we only need to consider single deviations from silence. Assume that Alice is the active player. Since Bob never says anything, there is no change to her information and her search costs remain the same. By monotonicity, Bob's search cost can only decrease if she deviates. If it decreases enough, then by the previous lemma, he will choose to search and acquire Alice's object. Otherwise, payoffs are the same, so deviating can only reduce her payoff. ■

It is not possible for a player to get ripped off in equilibrium, because they can opt-out.

**Lemma 3** *In no equilibrium does only one player learn their counterpart's secret.*

**Proof.** If Alice surrenders her secret without learning Bob's, her payoff is  $-\alpha_A$  which is worse than her abort payoff of 0. So Alice would deviate from any such equilibrium by aborting at the start. Similarly, Bob would abort at his first chance. ■

Since Alice knows anything that Bob tries to compute, computation is wasteful. Instead, an efficient equilibrium requires that Alice and Bob tell each other their entire secrets. This raises the question: when would Alice prefer to abort the conversation rather than complete it? If Bob's computational cost is sufficiently low that he would guess Alice's secret anyway, no matter what she does, then she cannot prevent Bob from learning her secret and she has nothing to gain by aborting the conversation. In this situation we will say that she is committed to the transaction in the sense that she has lost control of her own object and it is (weakly) dominant for her to continue the conversation. We define commitment regions for Alice and Bob to be the sets  $X_A = [0, \infty) \times [0, \alpha_B]$  and  $X_B = [0, \beta_A] \times [0, \infty)$ .

Alice's position is at risk if she is committed to the transaction but Bob is not. In Alice's unilateral commitment region  $X_A - X_B$ , Bob can walk away with both objects, leaving her with nothing. Whether he would do so depends upon his search cost. We define  $Y_A = [\beta_A, \infty) \times [0, \beta_B]$  and  $Y_B = [0, \alpha_A] \times [\alpha_B, \infty)$  to be their danger regions. At a point in Alice's danger region  $Y_A$ , Alice is committed to the transaction but Bob is not; furthermore, it is clearly in his interest to abort immediately since the benefit of getting Alice's object and not losing his (even after paying the computation costs that he will incur) exceed the potential gains from trade (see Figure 2).

**Lemma 4** *If the conversation enters Alice's danger region  $Y_A$  then she will lose her object but Bob will not lose his.*

**Proof.** We note first that if a conversation enters Alice's danger region then it will never leave it. Consider a conversation  $h$  that terminates in  $Y_A$  and a conversation  $h'$  that extends  $h$  and terminates outside  $Y_A$ . We consider Bob's behaviour in the games  $\Gamma(h)$  and  $\Gamma(h')$ . At  $h$  Alice is committed to the transaction but Bob is not, so his payoff would be  $\alpha_B - \bar{c}_B$ . The conversation can only move out of  $Y_A$  by moving into Bob's commitment region, so at  $h'$  both agents are committed to the transaction and Bob's payoff will be  $\alpha_B - \beta_B - \bar{c}'_B$ , which is strictly less than  $\alpha_B - c_B$  since, by monotonicity,  $c_B < \beta_B + \bar{c}'_B$ . Thus Bob's payoff will be strictly lower if the conversation were to leave  $Y_A$ . This could only happen by his sending a message that committed him to the exchange, which he would never do. Since the conversation must eventually terminate (there is only a finite number of messages to be sent), it must terminate in  $Y_A$ . The result then follows. ■

As a consequence, we have:

**Lemma 5** *Every equilibrium cost sequence must avoid the danger regions  $Y_A$  and  $Y_B$ .*

Let  $Z_A = X_A - X_B - Y_A$  and  $Z_B = X_B - X_A - Y_B$  be the agents' safe unilateral commitment regions (see Figure 2).

**Lemma 6** *Every complete equilibrium cost sequence must enter a safe unilateral commitment region, either  $Z_A$  or  $Z_B$ .*

**Proof.** If  $c = (c_A, c_B)$  is the complete equilibrium cost sequence associated with an equilibrium then at the beginning, when costs are  $(c_A^0, c_B^0)$ , neither agent is committed to the exchange. At the end, when costs are  $(0, 0)$ , both are committed. Since the agents move alternately, at some point one agent must be committed while the other remains uncommitted. Moreover, this point lies outside the danger regions by Lemma 5, and therefore lies inside a unilateral commitment region. ■

**Theorem 1** *A complete cost sequence  $c = (c_A, c_B)$  is supported in equilibrium if and only if it avoids the danger regions.*

**Proof.** The necessity of avoiding danger regions is clear from Lemma 5. It is also a sufficient condition. The guessing strategy is completely determined by Lemma 1. On the equilibrium path the message strategy is determined by the cost sequence (we require, as discussed in relation to Definition 1 that agents randomise between possible messages consistent with the specified cost sequence). Off the equilibrium path, we fix the message rules to silence. In these subgames, silence is an equilibrium by Lemma 2.

We now check for profitable deviations on the equilibrium path. Since silence is played after any deviation from the cost sequence, all deviations terminate the exchange of messages. By Lemma 6, the cost sequence goes through three stages when neither player is committed, one player is committed, and both are committed. If Alice terminates the exchange before either player is committed, then she does not learn Bob's secret, and her deviation payoff is  $0 < \beta_A - \alpha_A$ . If she terminates the exchange when only she is committed, then she loses her object without gaining Bob's, so her payoff is  $-\alpha_A$ . If she terminates when only Bob is committed, then she guesses Bob's secret without revealing her own for a payoff  $\beta_A - \bar{c}_A < \beta_A - \alpha_A$ . Finally, if she terminates after both players are committed, then both players learn each others secrets after some computation, so Alice's payoff is  $\beta_A - \alpha_A - \bar{c}_A$ . ■

If there are any grid points available in a safe unilateral commitment region, then trade can be implemented.

**Theorem 2** *There is a trade equilibrium if and only if the players' safe unilateral commitment region  $Z_A \cup Z_B$  overlaps with the grid  $G$  of feasible computation costs.*

**Corollary 3** *If trade can be implemented, then it can be implemented in three steps. In particular, if  $(\tilde{c}_A, \tilde{c}_B)$  lies in Alice's commitment region, then the 3-turn cost sequence*

$$\begin{aligned} c_A &= (c_A^0, 0) \\ c_B &= (c_B^0, \tilde{c}_B, 0) \end{aligned}$$

*is supported in equilibrium. In this equilibrium Alice speaks twice, Bob only once.*

We now show how this analysis can be modified if we drop Assumption 6 that all messages are permissible. This assumption is used in two ways. Firstly, since the information value of messages depends only on their size, not on their content, it allows us to use  $\mathbb{N} \times \mathbb{N}$  as the message space. This is convenient but immaterial. We can just as easily use  $\mathcal{M} \times \mathcal{M}$ , which has all the order properties that we need, and allow any messages, not just permissible messages. All the steps of the argument still work. In particular the key monotonicity Lemma, which is used in the proof of both Lemma 2 and Theorem 1, remains valid. The other way that it is used is to ensure that there are enough grid points: that the computational grid is sufficiently dense that it will overlap the safe unilateral commitment region  $Z_A \cup Z_B$  and support an exchange. If  $G_0$  meets  $Z_A \cup Z_B$  then  $G$  will meet  $Z_A \cup Z_B$  since  $G_0 \subset G$ .

**Assumption 7 (Enough permissible points)** *The permissible cost grid  $G_0$  meets  $Z_A \cup Z_B$ .*

We can now restate the main results with this weaker assumption.

**Theorem 3** *A complete cost sequence  $c = (c_A, c_B)$  is supported in equilibrium if and only if it avoids the danger regions.*

**Theorem 4** *There is a trade equilibrium if there are enough permissible points. The equilibrium can be supported by a complete cost sequence.*

The protocol that we will develop below will provide a linear grid of permissible points that can be made as fine as desired.

## 4.2 Discussion

There are thus three distinct phases in the exchange. Initially neither player is committed, with the initial stages of the game occurring outside  $X_A \cup X_B$ . The game could be aborted at this stage with no loss to either player. Eventually one player makes a safe unilateral commitment and play enters  $Z_A \cup Z_B$ . It then progresses to a stage of mutual commitment  $X_A \cap X_B$ . There is an intrinsic indivisibility here, into no matter how many steps the transaction is divided. The crux is the moment when one player makes a unilateral commitment, which they will do only if they can do so safely.

In fact we notice that if exchange can be implemented, then it can be done very simply in a three step exchange. This emphasises the fact that the apparent divisibility introduced by proceeding in steps does not buy us as much as may appear at first glance. It is interesting to note that in principle this requires only one asset to be made divisible. This three-turn secret exchange protocol is similar to the Jakobsson (1995) note-ripping exchange. If Alice wants to trade a \$100 note for Bob's television, she could tear her \$100 note in half. She would then give Bob one half, promising the second half on receipt of the television. The main difference between the protocols is the behaviour off the equilibrium path. In our three-turn secret exchange protocol, if Alice does not surrender her secret, then Bob learns it (at considerable cost). In the note-ripping exchange, if Alice keeps the second half of the note, then the note is worthless and is effectively destroyed.

The Blum-Damgard protocol is a special case of our set up. In their protocol a single bit of information is exchanged at each step, and the expected search cost halves each time. This provides a rather coarse, exponentially spaced exchange grid. Whether this supports exchange depends on whether it provides enough points to allow one player to commit safely, and for many reasonable parameter values it will not do so. In that case, the exchange path passes through one player's danger region. If reached, that player would lose everything, so they would prefer to abort at the beginning of the exchange.

We recall that, as discussed in Section 1, we assume that the agents are committed to the exchange mechanism as designed. That is, they can either play it out or not, according to their best interest, but following the rules of the game. They cannot stop and redesign the game or renegotiate the terms of the exchange. In particular, if one player is indifferent between continuing or not (for example, in transferring the last bit of information), but continuation is valuable to the other, the first player cannot hold the other hostage and demand a fee in order to continue. In our environment it is hard to see how such a hold-up could be implemented, since it would require a further exchange (of the bribe in return for continuation) that would be equally

difficult to implement<sup>17</sup>. On this point we refer to the incomplete contracts literature (see for example Pitchford & Snyder (2004)), to which we have nothing to add.

We now turn from the simplistic full credibility model, and show that the basic intuition flows through into a more realistic environment. There are two main tasks. The first is to show that we can replace full credibility by  $\varepsilon$ -credibility, supported by probabilistic proofs; this is done in Section 4.3. The other is to display, for any assets  $\alpha, \beta$ , a zero knowledge protocol that provides enough permissible points to support exchange; this is done in Section 5.

### 4.3 Exchange with probabilistic proofs

We now consider the case where players' assertions may not be fully credible. They may offer evidence of their honesty, through their willingness to participate in probabilistic proof protocols, but there is always a chance that they have cheated and have not been detected. That is, their assertions are only  $\varepsilon$ -credible, not fully credible. In this section we will show that this is sufficient, provided  $\varepsilon$  is sufficiently small.

The main complication that  $\varepsilon$ -credibility introduces is that expected costs (which now of course depend upon beliefs) may no longer be monotonic. In the full credibility game, any message  $\tau$  from Alice can only decrease the size of the set  $\mathbb{S}_\tau$  where Bob believes her secret to be hidden. Once we admit doubt about Alice's initial assertions as to how she has encoded her secret it is possible that Bob may judge her message  $\tau$  to be implausible, in the sense that he no longer believes any of her previous assertions. In this case the set of possible hiding places  $\mathbb{S}_\tau$  may expand rather than contract. It is not clear that Alice would not want to send such a message, even if she could avoid it. If Bob were at a history where he is committed to searching for Alice's secret, then Alice might wish deter him from searching by sending a message that he found implausible, causing him to doubt the information on which he had been relying.

We formalise the game, building on the previous structure. Before the game starts Alice and Bob know their secrets  $s_A$  and  $s_B$ , which are uniformly randomly and independently distributed in  $\mathbb{S}$ . The game proceeds in stages. First they simultaneously and independently choose decompositions  $\sigma_A \in \mathbb{S}^N$  and  $\sigma_B \in \mathbb{S}^N$  of their secrets. They commit to these choices by exchanging encrypted versions. Alice asserts that  $(s_A, \sigma_A) \in \Pi \subset \Omega = \mathbb{S} \times \mathbb{S}^N$  and Bob that  $(s_B, \sigma_B) \in \Pi \subset \Omega = \mathbb{S} \times \mathbb{S}^N$ , but there is no way for the other party to know whether they have set up honestly or whether they are lying. We will say that they have set up honestly if these assertions are in fact true. It will be convenient below, when describing information structures, to distinguish notationally between Alice's copy  $\Omega^A$  and Bob's copy  $\Omega^B$  of  $\Omega = \mathbb{S} \times \mathbb{S}^N$ .

They then choose strategies and play a probabilistic proof game designed to demonstrate their honesty; the precise game and the strategy space will be specified in the cryptography section below. Let  $\zeta$  be a transcript of the messages sent in the proof game. The outcome of the proof game is either a success or failure for each agent. If Alice is successful we will say that she has convinced Bob that she has set up honestly. She can always do so if she has indeed set up honestly, but she will fail to convince Bob with probability at least  $1 - \varepsilon_A$  if she has not been honest ( $\varepsilon_A > 0$  is a parameter fixed in advance). If she has been honest, then not only can she convince Bob but no useful information, beyond the fact that she has been able to succeed in the proof game, leaks out. Similar assertions apply to Bob.

Since the players do not observe each other's choices we may model the set up phase of the exchange as a simultaneous choice: Alice chooses a decomposition  $\sigma_A$  and her strategy in the proof game, and Bob simultaneously chooses  $\sigma_B$  and his proof strategy. A history of the set up phase is thus of the form  $(\sigma, \zeta)$  where  $\sigma = (\sigma_A, \sigma_B)$  and  $\zeta$  is a transcript of the proof game.  $\zeta$

<sup>17</sup>There is no infrastructure, for example an asset register providing appropriate services, to facilitate such an exchange. There is also the possibility of an infinite regress, if the hold up were in turn held up ...

is publicly known, as is the outcome: whether Alice and Bob have been successful in convincing the other party of their honesty.

They then play a conversation game, sending messages that open envelopes and reveal pieces of their secrets. A conversation will be a pair  $\tau = (\tau_A, \tau_B)$ , where  $\tau_A = (\tau_A^1 \subset \tau_A^2 \subset \tau_A^3 \subset \dots \subset \tau_A^k)$  and  $\tau_B = (\tau_B^1 \subset \tau_B^2 \subset \tau_B^3 \subset \dots \subset \tau_B^l)$  are increasing sequence of messages  $\tau_A^i \subset \sigma_A$  and  $\tau_B^i \subset \sigma_B$  sent by Alice and Bob respectively.<sup>18</sup> The agents must provide new information each time they speak, or the conversation terminates. So we require that  $\tau_A^1 \neq \emptyset$  unless  $k = 1$  and that all the inclusions  $\tau_A^{i-1} \subset \tau_A^i$  be strict except possibly for the last one. If  $k = 1$  and  $\tau_A^k = \emptyset$ , or if  $\tau_A^{k-1} = \tau_A^k$  then we will say that the message  $\tau_A^k$  is *terminal*: it is Alice's turn to speak but she has declined to reveal any more information. Such a message terminates the whole conversation. We will write  $\bar{\tau}_A = \tau_A^k$  for the biggest message that Alice sends in the conversation — it includes all earlier messages. We make corresponding definitions for  $\tau_B$ . A conversation thus defines an increasing zigzag path  $((\tau_A^0, \tau_B^0), (\tau_A^1, \tau_B^0), (\tau_A^1, \tau_B^1), (\tau_A^2, \tau_B^1), \dots, (\bar{\tau}_A, \bar{\tau}_B))$  in  $\mathcal{M}_A \times \mathcal{M}_B$ . Here we write  $(\tau_A^0, \tau_B^0) = (\emptyset, \emptyset)$  for the initial state before anyone speaks. When the conversation terminates, as it must eventually do, the agents choose strategies  $\gamma = (\gamma_A, \gamma_B)$  in the search game  $\Gamma$  as in the full credibility game.

A history is thus a sequence of the form  $((\sigma, \zeta), \tau, \gamma)$  or an initial segment of such a sequence, where  $\sigma$  is an initial decomposition of the players' secrets,  $\zeta$  is a history in the proof game,  $\tau$  is a conversation history, and  $\gamma$  is a search strategy profile. We require that  $\tau$  be a terminal conversation if  $\gamma \neq \emptyset$ . We will label the players' information sets by  $h = (\zeta, \tau)$ , the *publicly observable* part of the history. We write  $h = * = \emptyset$  for the initial state. We write  $h = \zeta$  rather than  $(\zeta, \emptyset)$  if  $\tau = \emptyset$ .

We let  $H$  be the set of all such observable histories. At any  $h \in H$  Alice knows her own secret  $s_A$  and any decomposition  $\sigma_A$  that she has chosen, as well as the public history  $h$ , but she does not know  $(s_B, \sigma_B) \in \Omega^B$  so her information set is isomorphic to  $\Omega^B$ . If we identify her information set with  $\{h\} \times \Omega^B$  then Alice's view of the state of the game is parametrised by  $H \times \Omega^B$ . Her beliefs at  $h$  are given by a probability distribution  $\mu_A(h)$  on  $\Omega^B$ . Similarly, Bob's view of the state of the game is parametrised by  $H \times \Omega^A$ . The information sets that are active in the game are those associated with the active player: at a non-initial, non-terminal history  $h$  the information set is  $\{h\} \times \Omega^B$  if it is Alice's turn to move,  $\{h\} \times \Omega^A$  if it is Bob's. At an initial or terminal node the players move simultaneously and there will be two information sets (**one for each agent**).

To summarise, a strategy for Alice is a choice of action at every information set  $h$  where it is her turn to move:

1. At  $h = *$  she must choose  $\sigma_A \in \mathbb{S}^N$  and a strategy in the proof game
2. At a non-terminal history  $h = (\zeta, \tau)$  she must choose a message  $\tilde{\tau}$  such that  $\bar{\tau}_A \subset \tilde{\tau} \subset \sigma_A$
3. At a terminal history  $h$  she must choose a search strategy  $\gamma_A$  (including not searching) in the search game  $\Gamma(h)$ .

We now consider Alice's beliefs in more detail. To reduce notation we will drop the superscript and write  $\Omega$  rather than  $\Omega^B$  when discussing her information sets. We refer back to Section 4 for definitions of  $\pi : \Pi \rightarrow \mathbb{S}$ ,  $\Omega_\tau$ ,  $\Pi_\tau$ , and  $\mathbb{S}_\tau$ , which we write  $\mathbb{S}_{\Pi, \tau}$  if  $\Pi$  needs to be made explicit. Recall that  $\mathbb{S}_{\Pi, \tau}$  is the image of  $\Pi_\tau$  under  $\pi$ , and that  $\mathbb{S} = \mathbb{S}_{\Omega, \tau}$  is the image of  $\Omega_\tau$  under  $\pi$ . We will say that a message  $\tau$  is *implausible* if  $\mathbb{S}_{\Pi, \tau} = \emptyset$ . That is, there exists no secret  $s' \in \mathbb{S}$  compatible with the message  $\tau$ . Note that this is a property of the message, not of Alice's secret

<sup>18</sup>We require the messages be monotonic increasing for notational convenience. In practice, Alice need not resend information she has already given Bob.

$s_A$ . We will say that a history  $h = (\zeta, \tau)$  is implausible (to Alice) if either the proof  $\zeta$  failed to convince her that Bob set up honestly or if the conversation  $\tau$  contains an implausible message from Bob. If  $h$  is an implausible history then any extension of  $h$  remains implausible.

**Assumption 8** *The plausibility of any history  $h = (\zeta, \tau)$  with respect to  $\Pi$  is immediately obvious from its form.*

Alice is naturally concerned that Bob may have been deceptive in the way that he has encoded his secret. As discussed in Section 3, we need to be careful about how much computational power Alice can bring to the updating of her beliefs. If she were allowed unlimited computation, she could attempt to guess Bob's secret, and then update her beliefs based on information she calculated. We therefore apply Assumption 2 (no side calculations), and declare that she continues to believe that Bob's secret is uniformly distributed in the range of values that can be inferred from the envelopes he has opened (and uniformly distributed on the whole range if she finds that he has cheated).

**Definition 3 (sceptical beliefs)** *We will say that Alice has sceptical beliefs if*

1. *Initially, she believes that  $(s, \sigma)$  is distributed uniformly in  $\Pi$*
2. *After any plausible (to Alice) public history  $h = (\zeta, \tau)$  she believes that  $(s, \sigma)$  is distributed uniformly in  $\Pi_{\bar{\tau}_B}$ , and hence that  $s$  is distributed uniformly in  $\mathbb{S}_{\Pi, \bar{\tau}_B}$ ; here  $\bar{\tau}_B$  is the last message sent by Bob*
3. *After any implausible history  $h = (\zeta, \tau)$  she believes that  $(s, \sigma)$  is distributed uniformly in  $\Omega_{\bar{\tau}_B}$ , and hence that  $s$  is distributed uniformly in  $\mathbb{S} = \mathbb{S}_{\Omega, \bar{\tau}_B}$*

Thus initially she believes Bob's assertion that  $(s, \sigma) \in \Pi$ , updating her belief using Bayes rule after plausible messages. As soon as he fails to act plausibly she believes nothing that he has said that she cannot check herself: she revises her initial belief to  $(s, \sigma) \in \Omega$ , updating it only with what she can herself verify in Bob's messages.

**Lemma 7** *If Bob's strategy is honest (that is, he sets up honestly, and chooses a proof strategy that is guaranteed to convince Alice if he is in fact honest), and Alice has sceptical beliefs, then those beliefs are Bayesian.*

**Proof.** Alice's updating strategy is explicitly Bayesian, except when she receives the first implausible message (either an unconvincing proof  $\zeta$  or an implausible message  $\tau$  in the conversation stage). But receiving such a message is a zero probability event. Under our assumption about Bob's proof strategy she can receive an unconvincing proof only if  $(s_B, \sigma_B) \notin \Pi$ . Likewise she can receive an implausible message  $\tau$  only if  $(s_B, \sigma_B) \notin \Pi$ . But by the assumption about Alice's beliefs, prior to receiving the first implausible message she believes with probability 1 that  $(s_B, \sigma_B) \in \Pi$ . So receiving a first implausible message is a zero probability event. ■

If agents have sceptical beliefs then expected search costs at  $h = (\zeta, \tau)$  in the  $\varepsilon$ -credibility game are easily calculated and they are closely related to expected costs at  $h = \tau$  in the full credibility game (recall that  $\bar{\tau}_B$  is the final message received from Bob in the conversation  $\tau$ )

$$c_A(\tau) = \begin{cases} |\mathbb{S}_{\bar{\tau}_B}| - 1 & \text{if } \tau \text{ is plausible} \\ |\mathbb{S}| - 1 & \text{if } \tau \text{ is implausible.} \end{cases}$$

**Corollary 4** *Given sceptical beliefs, expected search costs at plausible histories are exactly as in the underlying full credibility game.*

**Corollary 5** *Given sceptical beliefs, expected search costs are monotone along plausible histories.*

Given the observable history  $h = (\zeta, \tau) \in H$ , we call  $c(h) = (c_A, c_B) = ((c_A^1, c_A^2, c_A^3, \dots, c_A^k), (c_B^1, c_B^2, c_B^3, \dots, c_B^l))$ , where  $c_A^i(\tau) = |\mathbb{S}_\tau| - 1$  if  $\tau$  is plausible and  $c_A^i(\tau) = |\mathbb{S}| - 1$  if  $\tau$  is implausible, the cost sequence induced by  $h$  given sceptical beliefs. Note that  $c(h)$  is not monotone unless  $h$  is plausible. We will say that  $h$  is a complete history if the cost sequence  $c(h)$  is complete. Note that any complete history must be plausible.

**Theorem 5** *Suppose that players have sceptical beliefs and that each player's strength of evidence is  $\varepsilon_A < 1 - \frac{\alpha_A}{\beta_A}$  and  $\varepsilon_B < 1 - \frac{\beta_B}{\alpha_B}$ . Then there exists a sceptical belief perfect Bayes-Nash equilibrium that induces the complete computation cost sequence  $c = (c_A, c_B)$  in the cost grid  $G$  if and only if  $c$  avoids the danger regions.*

**Proof.** Assume that  $h$  is the observable part of a perfect Bayes Nash equilibrium in the  $\varepsilon$ -credibility game, that  $h$  is complete, but that  $c = c(h)$  encounters a danger region. Then, by the proof of Theorem 1, it would be a strictly profitable deviation in the full credibility game for one of the agents to abort the exchange before it enters the danger region. But the strategy of aborting is available in the  $\varepsilon$ -game as well. Since  $h$  is plausible the payoffs to continuing or aborting are exactly as in the full credibility game. So  $h$  cannot be part of a perfect equilibrium.

Now assume that we are given a complete cost sequence  $c$  in  $G$  that avoids the danger region. By Theorem 1  $c$  is supported by a perfect equilibrium in the full credibility game. We construct an equilibrium in the  $\varepsilon$ -game as follows. We describe the strategy from Alice's point of view. Bob's strategy is similar. If Alice has been dishonest she may for a time be able to conceal that fact. We will say that she conceals her type if she sends a message  $\tau$  such that  $\mathbb{S}_\tau \neq \emptyset$ , and that she reveals her type if she sends a message  $\tau$  such that  $\mathbb{S}_\tau = \emptyset$ .

1. Her beliefs are sceptical
2. She sets up honestly and chooses a proof strategy that is guaranteed, if she has been honest, to convince Bob
3. If Bob has acted implausibly then she believes that she can never acquire his object. The best that she can do is to discourage him from searching for hers (even if she has set up dishonestly, Bob might stumble upon her object if he does any searching). So she will reveal her type if she can (she can always do so if she has set up dishonestly); otherwise she will abort and not search.
4. We now consider histories  $h = (\zeta, \tau)$  where Bob is plausible. If Alice is implausible then she has already discouraged Bob from searching and she can send no messages that he will believe, so she aborts any conversation and plays the search strategy recommended at  $\tau$  in the full credibility game.
5. If Alice is honest and plausible, she follows the recommendation at  $\tau$  in the full credibility game (she can always do so and will remain plausible).
6. If Alice is dishonest but remains plausible, then she will reveal her type if she can (in order to discourage Bob from searching), rather than to abort or to move to a node where Bob will abort. Otherwise she will follow the recommendation at  $\tau$  in the full credibility game, concealing her type if possible, in order to encourage Bob to reveal more information.

It remains to specify how Alice will choose her messages if she is following the recommendations of the full credibility game. Among the messages that Alice can send, let  $\tau_{pl}$  be a plausible

message of maximal length (such a message clearly exists; it may be the empty message). If there is more than one such, she chooses  $\tau_{pl}$  from amongst them randomly. Any message  $\tau \subset \tau_{pl}$  is plausible, and choosing such messages delays as long as possible running out of plausible messages that can be sent. So we can assume that Alice randomly chooses any message of the required length from within  $\tau_{pl}$  if she wishes to conceal her type, and any implausible message (if such exist) if she wishes to reveal her type.

We note, by Lemma 7, that beliefs are Bayesian. To show that strategies are rational it suffices to show that there is no strictly profitable one step deviation. We focus on the key decision of whether or not to set up honestly (the remaining possible deviations are routine to check). If Alice is honest, then the exchange will be implemented costlessly, giving her an outcome worth  $\beta_A - \alpha_A$ . If she is dishonest, then the best she could do would be to acquire both objects at zero cost, giving her an outcome  $\beta_A$ . But she can achieve this only if she can fake her way through the proof game, which she can do with probability at most  $\varepsilon_A$ ; otherwise the exchange will fail, leaving her with outcome 0. So her expected gain from deviating from truthfulness is at most  $\varepsilon_A \beta_A + (1 - \varepsilon_A) 0$  which is strictly less than  $\beta_A - \alpha_A$  under our assumption on  $\varepsilon_A$ . ■

## 5 The Cryptographic Protocol

In this section we show how Alice can divide up her secret and release it gradually to Bob, using a construction suggested by Schoenmakers (2005). Recall that we wrote  $G = \{n\delta : n \in \mathbb{N}\}$  for the grid with mesh  $\delta$  consisting of the non-negative integral multiples of  $\delta$ , a parameter fixed before the beginning of the protocol. Alice will send a series of messages so that, at every point in the protocol, Bob's expected cost for guessing Alice's secret is in  $G$ .

Alice finds the unique quotient  $Q$  and remainder  $b \in [0, \delta)$  such that

$$s_A = \delta Q + b$$

She gradually reveals information about her secret by opening to Bob a series of commitments. Each commitment, when opened, reveals that  $Q$  is not equal to some particular value. Finally, when only one possible value for  $Q$  remains, she also reveals  $b$ . If the exchange terminates, Bob can compute Alice's secret by testing each of the remaining candidates for  $s_A$  in random order. This is an optimal search strategy provided that Bob believes every candidate is equally likely. If Alice told the truth so that the true secret is among the candidates, then Bob tests half of the candidates on average before discovering Alice's secret. Otherwise, he would try all candidates before discovering Alice lied. We assume that it costs Bob  $t$  to test if a candidate secret  $s'$  is the true secret by testing  $[s'] = S$ . If Bob believes Alice split her secret up honestly, and Bob considers there are  $n - k$  remaining possible values for  $Q$ , then Bob expects computing Alice's secret would cost him  $(n - k)t\delta/2$ . This derivation of the computational cost requires that Bob believe all remaining candidates are equally likely. When Alice reveals  $Q \neq 0$ , Bob could potentially learn more about  $s$  than  $s \notin [0, \delta)$ . For example, if Alice only chooses to reveal  $Q \neq 0$  first when  $Q = 1$ , then Bob would conclude that  $s \in [\delta, 2\delta - 1)$ . A proof that our protocol does preserve the uniform distribution on possible values of  $s_A$  is contained in Appendix section A.2.5.

Of course, Bob should not blindly believe Alice's claims about the values of  $Q$ , so we make Alice commit to  $Q$  in advance (in a rather complicated way described below), then convince Bob with zero-knowledge proofs that she has committed to the right number and that her claims about its value are true.

Since  $Q$  is drawn from only a small range, it would be very easy for Bob to guess. To prevent Bob from guessing  $Q$  directly, we employ a different commitment scheme, known as Pederson commitments, described in Appendix A.1. A value committed to with this scheme cannot be

guessed with any amount of computation, even if the value is taken from a small range. This forces Bob to conduct his search for Alice’s main secret  $s_A$  instead.

We now describe how Alice and Bob exchange their secrets  $s_A$  and  $s_B$ , which are locked up in the publicly known numbers  $S_A = [s_A]$  and  $S_B = [s_B]$ . Alice and Bob convince each other that they know their secrets using Protocol 1.<sup>19</sup> Then they split their secrets into pieces, which they trade in turn. Bob and Alice can check that the other’s pieces combine properly. When Alice gives Bob a piece, she just opens some commitments which Bob can check. The protocol ends when the exchange is complete.

Here is the protocol in more detail. We show the secret decomposition protocol for Alice only. Bob’s is identical and should be executed immediately after Alice’s.

Recall from Section 3 that all computations are done in the group of integers modulo some large prime  $p$ , and that  $\mathbb{U} = \{1, 2, \dots, p - 1\}$ .

**Protocol 2 (Secret decomposition protocol)** *Common input:  $g, S_A = [s_A] \in \mathbb{Z}_p^*$  and  $\delta \in \mathbb{U}$ . Also the parameters of a Pedersen commitment scheme (see Appendix section A.1), using the same prime  $p$ .*

*Alice’s input:  $s_A \in \mathbb{U}$ .*

1. Alice convinces Bob she knows  $s_A$  using Protocol 1.

2. Alice commits to  $Q$  and  $b$  by sending Bob:

- a Pedersen commitment to  $b$ .
- a sequence of Pedersen commitments to values  $a_0, a_1, \dots, a_{n-1}$  where

$$a_i = \begin{cases} 1 & \text{if } i = Q \\ 0 & \text{otherwise} \end{cases}$$

3. Alice convinces Bob that she constructed her pieces correctly:

- Alice uses the proof of knowledge of discrete logs (see 1) to prove that her commitments are correctly constructed, that is that  $s_A = \delta \sum_{i=0}^{n-1} ia_i + b$ .
- Alice proves  $b \in [0, \delta)$  using the interval membership zero-knowledge proof from Appendix A.2.4.
- Alice proves for all  $i \in \{0, 1, \dots, n-1\}$  that  $a_i$  is either 1 or 0, using the bit-membership zero-knowledge proof from Appendix A.2.3.
- Alice proves that exactly one of the,  $a_i$  is 1, using the knowledge of discrete logs proof from Appendix A.2.1.

**Protocol 3 (Secret exchange protocol)** *After Alice and Bob have both decomposed their secrets and run Protocol 2 to prove that they have done so correctly, they begin to reveal information about them. Alice and Bob alternate revealing pieces. If Bob’s current expected cost is  $c_B \in G$  and  $c'_B \in \{c \in G : c < c_B\}$ , then Alice randomly chooses  $(c_B - c'_B)/\delta$  of her remaining commitments to an  $a_i$  that is 0, and opens these commitments.*

---

<sup>19</sup>This prevents an impostor from pretending to be Alice and tricking Bob into giving up his secret in exchange for nothing. This proof would still have a very small chance of succeeding even if the impostor didn’t really know Alice’s secret. This is essentially an authentication problem which we don’t study here. If an impostor pretending to be Alice tries to sell her house, he has some small probability of convincing Bob to give up his money. If he fails, he only has to pay some small computation and communication costs. To deny an impostor any incentive to do this, Bob should ensure that the costs associated with attempting to dupe him outweigh the expected benefits. He could raise the costs by demanding Alice solve a difficult computational puzzle.

Bob's expected cost to search for  $s$  becomes  $c'_B$ . Bob can similarly reduce Alice's search space. When a player has only one commitment to an  $a_i$  remaining, the committed value must be 1. It is unnecessary to open this commitment. Instead, the final step of each player is to open their commitment to  $b$ . This reveals the value committed to in  $s_A$  (respectively  $s_B$ ).

We need to show that, at every step of the exchange protocol, the only information that the agents gain from the decomposition and exchange protocols is that the other's secret lies in the revealed range. That is, we have to demonstrate the zero knowledge property. First we introduce some more terminology, in order to make precise the notion of what computations can be done in reasonable time with reasonable probability. Define the security parameter  $k$  to be the logarithm base 2 (*i.e.* the length when written in binary) of the "large prime"  $p$  that defines the size of the group in which we work. Both the computation times and the probabilities that we use throughout the paper are functions of  $k$ . We say that a computation can be completed in *polynomial time* if there is a polynomial  $P$  such that the computation, running on inputs of binary-length  $k$ , finishes within  $P(k)$  steps. We say that a function  $f$  is *negligible* if for all positive polynomials  $P$ , for all sufficiently large  $k$ ,  $f(k) < 1/P(k)$ .

**Lemma 8** *Suppose an agent has run protocol 2 followed by some fraction of protocol 3. Let  $s$  be the secret committed to in protocol 2, and let  $I$  be the set of indices of  $a_i$ 's so far revealed to be 0 in the exchange protocol. Then the protocol run so far constitutes a zero knowledge proof that  $s \notin \bigcup_{i \in I} [\delta i, \delta(i+1))$ .*

**Proof.**

**soundness** Suppose that Alice cheats and attempts to run the protocol with a secret  $s$  that is actually in some interval other than the one she is going to reveal. We will show that she succeeds only with negligible probability. Consider the first time that Alice is about to "reveal" something that is untrue, and let  $j$  be the index of the first false one.<sup>20</sup> Then  $s \in [\delta j, \delta(j+1))$ , but Alice will attempt to open her commitment to  $a_j$  as zero.

- If Alice committed to a value other than zero, then she cannot open the commitment as zero except with negligible probability.
- If Alice did commit to zero, then, except with negligible probability, Alice's pieces are "correct" initially in the sense that they satisfy all the bullet points in Step 3. (This follows from the soundness of the protocols used.) Then we have  $s = \delta \sum_{i=0}^{n-1} i a_i + b$ , where  $b \in [0, \delta)$  and exactly one of the  $a_i$  is one, with the rest being 0, including  $a_j$ . Hence  $s$  cannot be in the interval  $[\delta j, \delta(j+1))$ .

**completeness** It is obvious that an honest Alice succeeds in completing the protocol correctly.

**zero knowledge** A formal proof that this protocol is zero knowledge is contained in Appendix A.2.5. The informal argument is that the initial Pedersen commitments reveal no information, and the fact that a particular  $a_i$  is zero reveals only that the secret is not in the corresponding interval. Because the commitments to  $a_i$ 's are opened in a random order, no probabilistic information is revealed by the order in which they are revealed. All the other parts of the protocol are themselves zero-knowledge proofs, so the zero-knowledge property is preserved when they are composed in sequence.

■

---

<sup>20</sup>The analysis is the same if the first falsely opened value is actually  $b$

**Corollary 6** *Protocol 2 followed by Protocol 3 satisfies all the assumptions made in Section 4, except obviously Assumption 4 (full credibility). In particular,*

*If Bob is honest, then the computation cost depends (linearly) only on the number of pieces left.*

**Assumption 5** *The decomposition is uniquely determined, given  $\delta$ , by the value of  $s$ .*

**Assumption 7** : *The permissible cost grid can be made fine enough.*

**Assumption 8** : *The plausibility is obvious from any history.*

**The assumption of Theorem 5** : *for any  $\epsilon_A > 0$ , the protocols can be parameterised so that the probability that Bob detects all of Alice’s cheating in the set-up stage is at least  $1-\epsilon_A$ .*

**Proof.** We prove each property in turn.

This follows from the protocol being zero-knowledge, because it implies that Bob’s information at any point is equivalent to knowing only the set that the secret has been revealed to lie in.

**Assumption 5** This is obvious.

**Assumption 7** The protocol produces a cost grid of interval  $\delta$ , which can be as small as 2, and hence accommodate any nonempty commitment region.

**Assumption 8** : This is obvious.

**The assumption of Theorem 5** : This follows from the soundness of the zero knowledge proofs, which implies that if Alice cheats on a certain claim then each run of its proof fails with probability at least  $1/2$ , independent of other runs of the protocol. Hence the proofs of correct set up can be run  $-\log_2 \epsilon_A$  times to achieve the required level of confidence.

■

## 6 Conclusion

In this paper we inquire into the possibility of simultaneous exchange of assets in an asynchronous communication environment, and into the institutional infrastructure required to support such exchange. We assume a minimal legal system that links assets to entries in an asset register, links agents to assets via digital identities, gives value to assets by facilitating the enforcement of property rights, and facilitates the management of identities and transfer of ownership. We do not assume the existence of a trusted intermediary who can facilitate exchange.

? and Damgard (1995) proposed a well known cryptographic protocol for such exchanges. They suggested that ownership of assets be associated with knowledge of secrets, and that assets be made divisible by encrypting these secrets with long encryption keys. Exchange of assets may be achieved by bit-by-bit incremental exchange of keys, supported by cryptographic protocols and probabilistic zero knowledge proofs that allow safe and highly credible exchange of partial information. We provide a game theoretic analysis, for computationally bounded agents, of exchange protocols of this type.

Under an assumption that true communications are fully credible, we characterise protocols that support exchange as a subgame perfect equilibrium. We find that the divisibility intuition is somewhat misleading; anything that can be done incrementally can be done in three steps,

and we find that the Blum Damgard protocol can fail for reasonable parameter values. We show that if trade is individually rational then sufficient credibility to support exchange can always be established by cryptographic cheap talk. That is, anything that can be implemented in a full credibility world can be implemented using probabilistic proofs. Using this, we show that there do exist protocols that can implement any individually rational exchange of assets. To do so requires relatively fine grained manipulation of computational cost.

Thus the assumption of computationally bounded agents (an assumption of bounded rationality) usefully expands the set of institutions that can be implemented. It facilitates disaggregation and exchange of partial information, it allows agents to make useful types of commitment, and cryptographic cheap talk enlarges the range of credible communication.

## A Cryptographic Appendix

### A.1 Pedersen Commitments

In section 3 we argued that Alice can lock up a secret  $s$  with a one-way function like  $\exp_g$ . Bob would have to do a costly computation like trying every number in  $\mathbb{Z}_q$  until he finds her secret  $s$ . This argument depends on the assumption that the secret is drawn uniformly from  $\mathbb{Z}_q$  (from Bob's point of view). However, if  $s$  were not uniformly distributed, then Bob could focus his search on likely candidates. For example, if  $s \in \{0, 1\}$  then Bob would only have to check two candidates.

Pedersen's *parameterized scheme* (Pedersen 1991) allows Alice to safely lock up a secret drawn from a small set. Alice uses an extra random number  $r$ . Bob then needs to guess both the secret  $s$  and the random number  $r$  to open the commitment.

Let  $p$  and  $q$  be primes such that  $q|p-1$ . We need both primes to be large enough that exhaustively enumerating a set of that order is infeasible. (In practice  $q$  must be at least 160 bits). Let  $G(q)$  be the unique subgroup of  $\mathbb{Z}_p^*$  of order  $q$ . The commitment scheme requires two generators  $g$  and  $h$  of  $G(q)$ , chosen such that neither Alice nor Bob knows  $\log_g h$ . If Alice wants to commit to some secret  $s$ , she generates a random  $r$  and sends Bob the commitment

$$C(s, r) = g^s h^r \pmod{p}$$

Alice can honestly open her commitment by revealing  $(s, r)$ . She would find it difficult to dishonestly open her commitment to some other value  $(s', r')$  because this is at least as hard as computing  $\log_g h$ . If Alice knew  $(s, r)$  and  $(s', r')$ , then she could compute  $\log_g h$ :

$$\begin{aligned} g^s h^r &= g^{s'} h^{r'} \\ h^{r-r'} &= g^{s'-s} \\ h &= g^{(s'-s)/(r-r')} \\ \log_g h &= (s' - s)/(r - r'). \end{aligned}$$

We need to show how Alice and Bob can generate appropriate  $g$  and  $h$ . Note that  $x^q = 1 \pmod{p}$  iff  $x \in G(q)$ . This provides an easy test of whether an element  $x$  is in  $G(q)$ . Furthermore, since  $q$  is prime, every element of  $G(q)$  is also a generator of it. Alice and Bob can jointly construct a pair of such generators  $(g, h)$  as follows. First, Alice chooses  $u \in \mathbb{Z}_p^*$ , and sets  $g = u^{(p-1)/q}$ . By Euler's theorem, the order of  $g$  divides  $q$ . If  $g = 1$ , Alice tries again. Otherwise  $g$  has order  $q$ , and she tells Bob  $g$ . To generate  $h$ , they do the following:

- Bob generates a random  $b$  and sends Alice  $B = g^b \pmod{p}$  (without revealing  $b$ ),

- Alice generates a random  $a$  and sends Bob  $A = g^a \bmod p$  (without revealing  $a$ ),
- Both compute  $h = g^{ab} \bmod p$  (Bob computes  $A^b$  and Alice  $B^a$ ).

This is the Diffie-Hellman key exchange protocol (Diffie & Hellman 1976).

## A.2 Zero knowledge proofs

### A.2.1 Zero-knowledge proofs of knowledge of discrete logs

Since  $\exp_g$  is a one-way function, applying it to a secret  $s$  is like locking it up in a box. If the secret  $s$  is ever revealed, it is a trivial operation to test if the box contains the secret, but it is expensive to open the box by other means, such as trying out all possible secrets.

This following protocol allows a player to provide compelling evidence that they know how to open a locked-up secret without giving any hints on what the secret is. A dishonest prover that does not know the secret can successfully mimic an honest prover with probability  $\frac{1}{2}$ , but this number can be made arbitrarily small by repetition.

**Protocol 4 (Knowledge of discrete logs)** *We prove correctness of the proof of knowledge of discrete logs, Protocol 1, which we write out again more carefully here. Proves: Knowledge of  $s$  such that  $S = g^s \bmod p$ .*

*Common input:  $S, g$  in  $\mathbb{Z}_p^*$ . Also  $q$ , which is the order of  $g$  in  $\mathbb{Z}_p^*$ .*

*Prover's input:  $s \in \mathbb{Z}_q$ .*

*Repeat the following many times:*

1. *The prover chooses a random element  $r \in \mathbb{Z}_q$  and sends the verifier  $R = g^r \bmod p$ .*
2. *The verifier sends back a random challenge  $c \in \{0, 1\}$ .*
3. *The prover responds with  $y = r + cs \bmod q$ .*
4. *The verifier checks  $g^y = RS^c \bmod p$ .*

Note that if the verifier chooses the challenge  $c = 0$ , then the verifier expects to receive  $r$ , and if  $c = 1$ , they expect to receive  $r + s$ .

The protocol is complete, because the prover can always answer any challenge with knowledge of  $(r, s)$ . It is sound because a cheating prover must be able to provide a convincing answer in response to both  $c = 0$  and  $c = 1$  (if not, it has a  $1/2$  probability of being caught lying at each repetition). It can only do this by knowing  $s$  (assuming it can't compute discrete logs).

The proof is zero-knowledge, because the verifier could efficiently simulate the execution of the protocol alone, and hence does not get any help in computing anything about  $s$ . If the verifier chooses a challenge  $c = f(R)$ , then an efficient simulation would be:

1. Pick  $c \in \{0, 1\}$  stochastically so that  $P(c = 0) = P(f(U) = 0)$ , where  $U$  is uniformly distributed.
2. Pick  $r$  uniformly at random and output  $(R, c, r) = (g^r / S^c \bmod p, c, r)$ .

Clearly, the simulated output has the same distribution as the verifier's view of the protocol.<sup>21</sup>

Alice can use this protocol in Step 3 of Protocol 2 to prove that her commitments add up to the correct value. Recall that Alice has committed to a sequence of values  $a_0, a_1, \dots, a_{n-1}$ ,

<sup>21</sup>For a more formal treatment, see (Goldreich 2002)

$b$ , and Bob needs to be convinced that  $s = \delta \sum_{i=0}^{n-1} ia_i + b$ . Let  $A_0, A_1, \dots, A_{n-1}$  be (what Alice claims to be) the commitments to  $a_0, a_1, \dots, a_{n-1}$ , respectively, and let  $B$  be (what Alice claims to be) the commitment to  $b$ . Let

$$C = \left( \prod_{i=0 \dots n-1} A_i^{\delta i} \right) B.$$

(Bob can compute this for himself.) Then if Alice constructed  $A_0, \dots, A_{n-1}$  and  $B$  correctly,  $C$  is a commitment to  $\delta \sum_{i=0}^{n-1} ia_i + b$ , because for some value  $r$  (a function of the random values Alice used in her sequence of commitments, which she knows),

$$\begin{aligned} C &= g^{\delta \sum_{i=0}^{n-1} ia_i + b} h^r \\ &= S h^r \end{aligned}$$

So Alice can prove, using Protocol 1, that she knows the discrete log  $\log_h(C/S) \bmod p$ . We need to show that if Alice can prove she knows this, then she knows how to open the commitments  $\{S, C, A_i, B\}$  so that

1.  $A_0, \dots, A_{n-1}$  and  $B$  open so that  $C$  is a commitment to  $\delta \sum_{i=0}^{n-1} ia_i + b$ .
2.  $S$  and  $C$  open to the same value, and

and she does not know how to open  $\{S, C, A_i, B\}$  any other way.

If Alice knows how to open  $A_i$  and  $B$  (as she proved earlier in the protocol), then she knows how to open  $C$  so that it satisfies 1. Moreover, she does not know how to open  $C$  any other way – otherwise she could compute  $\log_g h \bmod p$  from the two different "openings" of  $C$ .

Now we will show that if Alice knows  $\log_h(C/S) \bmod p$ , then Alice knows how to open  $C$  and  $S$  to the same thing (namely  $s$ ). Recall that Alice knows  $(s, r)$  such that  $C = g^s h^r \bmod p$  and  $s'$  such that  $S = g^{s'} \bmod p$  (which she proved before). Let  $r' = \log_h(S/C)$ , which Alice supposedly knows. If  $r' = r$  then  $s = s'$ . If  $r \neq r'$  then Alice can open  $C$  in two different ways, to  $(s, r)$  and  $(s', r')$  and hence compute  $\log_h h \bmod p$ .

### A.2.2 Zero-knowledge proofs of disjunctions

We now consider the problem of aggregating two zero-knowledge proofs of propositions  $\varphi$  and  $\psi$ . Conjunction is trivial: Alice can prove  $\varphi \wedge \psi$  by proving  $\varphi$  and then  $\psi$ . Disjunction is more difficult. If Alice knows  $\varphi$  or  $\psi$  is true, she could just execute the zero-knowledge proof of the assertion she knows. However, this would not be a zero-knowledge proof of  $\varphi \vee \psi$ , as it reveals information beyond the veracity of  $\varphi \vee \psi$  – namely, which of  $\varphi$  or  $\psi$  is true.

(Cramer, Damgard & Schoenmakers 1994) prove  $\varphi \vee \psi$  by providing a genuine proof for one of the propositions, and a “fake” proof for the other. The prover is forced to answer a “difficult” challenge for any proposition, but can rig an “easy” challenge for the remaining proposition. The verifier only knows that one of the challenges was difficult, but does not know which one. The verifier concludes that at least one of the propositions is true. To use this this, the propositions must each have a zero knowledge proof with a single challenge that is supposed to be chosen uniformly by the verifier. We will assume for simplicity that the two challenges are taken from the set  $\{0, 1\}$ . A more general result (for boolean formulae of any fixed depth) is contained in (deSantis, di Crescenzo, Persiano & Yung 1994). *Each proof must follow this form:*

1. The prover chooses  $m^1$ , and sends it to the verifier.
2. The verifier picks a challenge  $c \in \{0, 1\}$  uniformly at random, and sends it to the prover.
3. The prover responds with  $m^2$ , based on  $(m^1, c)$ .
4. The verifier checks  $m^2$ , for which there is only one correct value for each  $(m^1, c)$ .

(For example, each round of the zero-knowledge proof of knowledge of discrete logarithms has this form.) Because the proof is zero knowledge, the prover can simulate it for any verifier, in particular an honest one. That is, for any proposition *including one it does not know to be true*, it can generate a transcript  $(m^1, c, m^2)$  that is identically distributed to the transcript of a successful interaction with an honest verifier.

This implies that the challenge  $c$  in the transcript is chosen uniformly and independently.

The disjunction protocol works as follows.

**Protocol 5 (Disjunctions)** *Proves:*  $\varphi_1 \vee \varphi_2$ .

*Common input:* the common inputs to  $\varphi_1$  and  $\varphi_2$ .

*Prover's input:* the prover's input for either the proof of  $\varphi_1$  or  $\varphi_2$  without loss of generality we assume the prover has the prover's input for  $\varphi_1$ .

1. If  $\varphi_1$  is true, then the prover chooses  $m_1^1$  as if it were proving only  $\varphi_1$ , and simulates a proof of  $\varphi_2$ , consisting of  $(m_2^1, c_2, m_2^2)$ .
2. The prover sends  $(m_1^1, m_2^1)$  to the verifier.
3. The verifier sends the prover a challenge  $c \in \{0, 1\}$  (supposed to be chosen uniformly and independently, but possibly not).
4. The prover sets  $c_1 = c + c_2 \bmod 2$ , then computes  $m_1^2$  honestly, as in the proof of  $\varphi_1$ .
5. The prover sends  $(c_1, c_2, m_1^2, m_2^2)$  to the verifier.
6. The verifier checks that  $c_1 + c_2 = c \bmod 2$  and checks for each  $i \in \{1, 2\}$  that the zero knowledge proof  $(m_i^1, c_i, m_i^2)$  is correct.

The protocol is obviously complete, because an honest prover can complete the proof of the disjunct that it knows to be true, and simulate the proof of the other. The protocol is sound, assuming that the two component protocols are sound, because a cheating prover who can succeed with probability  $p$  can cheat on at least one of the component protocols with probability at least  $p/2$ .

We now show why the protocol is zero-knowledge. In particular it does not reveal which of the propositions is true, no matter what program  $V^*$  the verifier runs. To establish this, we construct a simulation program that an uninformed verifier can run alone to generate identically distributed conversation transcripts:

1. Run the honest-verifier simulations  $M_1$  and  $M_2$  of the proofs of  $\varphi_1$  and  $\varphi_2$  respectively, which exist since protocols 1 and 2 are zero-knowledge. Call their outputs  $(m_1^1, c_1, m_1^2)$  and  $(m_2^1, c_2, m_2^2)$ . Because these are honest-verifier simulations,  $c_1$  and  $c_2$  are chosen uniformly and independently from  $\{0, 1\}$ .
2. If  $c_1 + c_2 = V^*(m_1^1, m_2^1) \bmod 2$ , then output  $(m_1^1, m_2^1), c_1 + c_2 \bmod 2, (c_1, c_2, m_1^2, m_2^2)$ .
3. Otherwise, fail with output  $\perp$ .

This simulator fails with probability at most  $1/2$ , since  $c_1 + c_2$  is uniformly distributed given  $(m_1^1, m_1^2)$ . As usual, the simulator can retry to decrease the failure probability arbitrarily.

Conditional on the simulator succeeding, its output is identically distributed to a conversation transcript between the prover and a verifier  $V^*$  (who may or may not be honest, that is, choose its challenge at random). To see this, again use the fact that  $c_1 + c_2 \bmod 2$  is uniformly distributed given  $(m_1^1, m_1^2)$ . This means that the probability of a simulation failing is  $1/2$ , independent of the message pair  $(m_1^1, m_1^2)$ . Hence the distribution of  $(m_1^1, m_1^2)$ , given the simulation is successful, is identical to the product distribution induced by doing both honest-verifier simulations independently. But this is exactly the distribution on  $(m_1^1, m_1^2)$  in a real interaction with  $V^*$ : one element is generated by the same simulation, and the other is correctly simulated by it. (Because each challenge  $c_1$  or  $c_2$  is uniformly and independently chosen given  $c_1 + c_2$ ). By construction, the distribution of  $c_1 + c_2 \bmod 2$  given  $(m_1^1, m_1^2)$  in successful simulations is exactly the same as in a real interaction with  $V^*$ . Finally, note that for each  $i \in \{1, 2\}$ , the acceptable value of  $m_i^2$  is uniquely determined given  $(m_i^1, c_i)$ . Hence the joint distribution on  $(m_1^1, m_1^2), c_1 + c_2 \bmod 2, (c_1, c_2, m_1^2, m_2^2)$  is identical to that of a real transcript of an interaction with (possibly dishonest) verifier  $V^*$ .

### A.2.3 Zero-knowledge proofs of a bit

This section describes (Mao 1998)'s zero-knowledge proof of knowledge that the prover knows how to open the parameterized commitment  $S = g^s h^u \bmod p$  so that the committed value  $s \in \{0, 1\}$  is a bit. This is the key building block in constructing zero-knowledge proofs of interval membership that  $s \in \{a, \dots, b\}$ . It is also a special case.

Our proof is based on (Mao 1998)'s proof, although our proof is easily shown to be zero-knowledge while Mao's proof is only known to be zero-knowledge if the verifier chooses challenges at random. Our proof is substantially less efficient, since it requires several rounds to achieve an acceptably low probability of allowing the prover to get away with cheating.

Notice that proving  $s \in \{0, 1\}$  is the same as proving  $(s = 0) \vee (s = 1)$ . We will describe how to adapt Protocol 4 for proving knowledge of discrete logs to proving  $s = 0$  and  $s = 1$ . We then apply the disjunction transformation from the previous section to construct a zero knowledge proof of  $(s = 0) \vee (s = 1)$ .

Alice can convince Bob that her commitment  $S = g^s h^u$  contains the secret is  $s = c$  by proving she knows  $u = \log_h(S/g^c)$  using Protocol 4 for proving knowledge of discrete logarithms. Alice can use this to open the commitment without revealing any information beyond the secret value, such as the secret parameter  $u$ .

This means Alice can prove  $s = 0$  by proving she knows  $\log_h S$  and  $s = 1$  by proving she knows  $\log_h(S/g)$ . When these two proofs are combined to prove  $s \in \{0, 1\}$ , they give the following protocol:

**Protocol 6 (Bit)** *Proves: knowledge of  $u$  such that  $S = g^s h^u \bmod p$  and  $s \in \{0, 1\}$ .*

*Common input:  $S, g, h \in G(q)$ .*

*Prover's input:  $u \in \mathbb{Z}_q$ .*

1. *The prover computes*

<i>if <math>s = 0</math></i>	<i>if <math>s = 1</math></i>
<i>randomly choose <math>r_0</math>, and set <math>R_0 = h^{r_0} \bmod p</math></i>	<i>randomly choose <math>r_1</math>, and set <math>R_1 = h^{r_1} \bmod p</math></i>
<i>randomly choose <math>c_1 \in \{0, 1\}</math> and <math>y_1 \in \mathbb{Z}_q</math></i>	<i>randomly choose <math>c_0 \in \{0, 1\}</math> and <math>y_0 \in \mathbb{Z}_q</math></i>
<i>set <math>R_1 = h^{y_1}/(S/g)^{c_1} \bmod p</math></i>	<i>set <math>R_0 = h^{y_0}/S^{c_0} \bmod p</math></i>
<i>and sends the verifier <math>(R_0, R_1)</math>.</i>	

2. The verifier sends the prover a random challenge,  $c \in \{0, 1\}$ .

3. The prover computes

$if\ s = 0$	$if\ s = 1$
$set\ c_0 = c - c_1 \bmod 2$	$set\ c_1 = c - c_0 \bmod 2$
$set\ y_0 = r_0 + c_0 s \bmod q$	$set\ y_1 = r_1 + c_1 s \bmod q$

and sends the verifier  $(c_0, y_0, c_1, y_1)$ .

4. The verifier checks that the challenges are valid  $c = c_0 + c_1 \bmod 2$ , and that the proofs that  $s = 0$  and  $s = 1$  are valid,

- $h^{y_0} = R_0 S^{c_0} \bmod p$ .
- $h^{y_1} = R_1 (S/g)^{c_1} \bmod p$ .

The complete, sound and zero-knowledge properties hold in this composite protocol, because they hold for the proofs of  $s = 0$  and  $s = 1$ , and the disjunction transformation preserves these properties.

#### A.2.4 Zero-knowledge proofs of interval membership

We now consider the problem of only *partially* opening a commitment, by revealing the secret lies in some interval  $s \in [x, x + l]$ . This allows the committer to reduce the computational cost to the receiver of forcing open the commitment by brute-force search. The committer controls this computational cost by controlling the size of the search space,  $l$ .

There are several standard zero-knowledge proofs of interval membership. The most efficient of these is (Boudot 2000), which under standard computational complexity assumptions requires only 2 kilobytes of information. We will present (Mao 1998)'s proof. (Boudot 2000) reports this protocol is 100 times less efficient than his own, but it is simpler to explain, and is efficient enough to be practical for our intended application.

The protocol we below proves that, for a given  $S = g^s \bmod p$  and a parameter  $n$ ,  $s \in [0, 2^n - 1]$ . To show that  $s \in [x, x + l]$ , we apply the proof twice: Let  $N$  be the length of  $l$  when written in binary. It suffices to prove that  $s - x \in [0, 2^N - 1]$  and  $l + x - s \in [0, 2^N - 1]$ . Commitments to both  $s - x$  and  $l + x - s$  can be generated using the homomorphism from section 3.1.

**Protocol 7 (Interval membership)** *Proves: knowledge of  $s \in [0, l] \subset \mathbb{Z}_q$  such that  $S = g^s \bmod p$ .*

*Common input:  $g, h, S \in G(q)$  and  $l, m_i \in \mathbb{Z}_q$ .*

*Prover's input:  $s \in G(q)$ .*

1. Let  $s_i$  be the  $i^{\text{th}}$  binary digit of  $s$ , so that  $s = \sum_{i=0}^n s_i 2^i$ .
2. The prover randomly chooses  $u_0, \dots, u_n \in \mathbb{Z}_q$ , and computes  $u = \sum_{i=0}^n u_i 2^i \bmod q$  and  $S_i = g^{s_i} h^{u_i} \bmod p$ . The prover sends each  $S_i$  to the verifier.
3. The verifier checks the equality of

$$Sh^u = \prod_{i=0}^n S_i^{2^i} \bmod p.$$

Notice that  $Sh^u = g^s h^u$ .

4. The prover runs the bit protocol to convince the verifier that she knows  $(s_i, u_i)$  such that  $s_i \in \{0, 1\}$  and  $S_i = g^{s_i} h^{u_i} \bmod p$ .
5. If the last two steps succeed, the verifier accepts the proof.

For completeness, notice that if the protocol is executed honestly, then

$$Sh^u = g^s h^u = g^{\sum_{i=0}^n s_i 2^i} h^{\sum_{i=0}^n u_i 2^i} = \prod_{i=0}^n g^{s_i 2^i} h^{u_i 2^i} = \prod_{i=0}^n S_i^{2^i} \bmod p.$$

For soundness, observe the product on the right side of the equation above is a commitment to  $\sum_{i=0}^n s_i 2^i$ , which equals the left expression which is a commitment to  $s$ . Since the prover can compute both sides of the equation, and the prover can not open any commitment two ways with non-negligible probability, it follows that  $s = \sum_{i=0}^n s_i 2^i$ . If the proofs that  $m_i \in \{0, 1\}$  succeeds, it follows that  $s \in [0, l]$ .

Zero-knowledge follows from the elementary properties of the individual steps.

### A.2.5 Proof of Lemma 8

Soundness and completeness are shown in the main body of the paper. We now want to show that protocol 2 followed by some fraction of protocol 3 is zero-knowledge. We want to show that, after Alice has opened some commitments, Bob learns only the implied restrictions on the possible values of Alice's secret  $s_A$ , and no other "leaked" information. It is important to show this for all partial steps of the protocol, not just the final stage, because Bob could use to his advantage any extra information he gleaned at any stage of the exchange protocol.

Let  $I$  be the set of indices  $i$  for which Alice has proven that  $s \notin [\delta i, \delta(i+1))$  (by opening a commitment to  $a_i = 0$ .) The special case  $I = \emptyset$  corresponds to running only the decomposition protocol, protocol 2; the special case where  $I$  contains every number in  $\{0, \dots, n-1\}$  except  $Q$  corresponds to running to the end of the exchange protocol, protocol 3. (The argument for the simulation of the final step, the revelation of  $b$ , is identical to that for the other commitments.) We show how a verifier can simulate the exchange knowing only that  $s \notin \bigcup_{i \in I} [\delta i, \delta(i+1))$ . The simulation works as follows:

1. For all  $i \in I$ , generate a commitment to zero.
2. For all  $i \notin I$ , generate a random element of the domain of Pedersen commitments. Also generate another random element (to be used to simulate a commitment to  $b$ ).
3. Simulate the secret decomposition protocol (Protocol 2) by using the commitments generated in Steps 1 and 2, then adopting the standard simulations for the zero knowledge proofs used in steps 1 and 3 of the secret decomposition protocol.
4. Simulate the secret exchange protocol (Protocol 3) by opening the commitments generated in step 1 in a random order.

We need to show that this generates a distribution identical to that of a real exchange. Recall that a Pedersen commitment to anything is a random element of the domain, so steps 1 and 2 generate exactly the same distribution on commitments as a real exchange. The simulations of the zero knowledge proofs used in steps 1 and 3 of 2 are shown above to be correct.

Finally, we need to show that the verifier learns nothing from the order the prover reveals the pieces  $\{a_i : i \in I\}$ . We do this by showing that the simulator chooses its order in a statistically identical manner. The prover randomly draws each successive piece without replacement from  $I$ .

This is equivalent to randomly choosing a permutation of  $I$  that specifies the revelation order. The simulator literally chooses a random permutation, so the piece orderings of the prover and the simulator are identically distributed.

## References

- Aumann, R. J. & Hart, S. (2003), ‘Long cheap talk’, *Econometrica* **71**(6), p1619.
- BenPorath, E. (2003), ‘Cheap talk in games with incomplete information’, *Journal of Economic Theory* **108**(1), p45.
- Blum, M. (1983a), ‘How to exchange (secret) keys’, *ACM Transactions on Computer Systems* **1**(2), 175–193.
- Boudot, F. (2000), Efficient proofs that a committed number lies in an interval, in B. Preneel, ed., ‘Advances in Cryptology, proc. EUROCRYPT 2000; Lecture notes in Computer Science Vol. 1807’, Springer.
- Buttayan, L. & Hubaux, J. (2001), Rational exchange: A formal model based on game theory, in ‘Proceedings of 2nd International Workshop on Electronic Commerce’.
- Buttayan, L., Hubaux, J. P. & Capkun, S. (2002), A formal analysis of syverson’s rational exchange protocol, in ‘Proceedings of the IEEE Computer Security’.
- Cramer, R., Damgard, I. & Schoenmakers, B. (1994), Proofs of partial knowledge and simplified design of witness hiding protocols, in ‘Proceedings of CRYPTO ’94’, Springer-Verlag, pp. 174–187.
- Damgard, I. B. (1995), ‘Practical and provably secure release of a secret and exchange of signatures’, *Journal of Cryptology* **8**(4), 201–222.
- Davis, M. D., Weyuker, E. J. & Sigal, R. (1994), *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*, Morgan Kaufmann.
- deSantis, A., di Crescenzo, G., Persiano, G. & Yung, M. (1994), On monotone formula closure of  $\text{szk}$ , in ‘IEEE Symposium on Foundations of Computer Science’, pp. 454–465.
- Diffie, W. & Hellman, M. (1976), ‘New directions in cryptography’, **IT-22 NUMBER=6**, 644–654.
- Dodis, Y. & Rabin, T. (2007), Cryptography and game theory, in N. Nisan, T. Roughgarden, E. Tardos & V. Vazirani, eds, ‘Algorithmic Game Theory’, Cambridge University Press, New York, pp. 181–205.
- Forges, F. (1990), ‘Universal mechanisms’, *Econometrica* **58**(6), 1341–1364.
- Forges, F. & Koessler, F. (2005), ‘Communication equilibria with partially verifiable types’, *Journal of Mathematical Economics* **41**(7), 793–811.
- Forges, F. & Koessler, F. (2006), ‘Long persuasion games’.
- Gerardi, D. (2004), ‘Unmediated communication in games with complete and incomplete information’, *Journal of Economic Theory* **114**(1), p104.

- Goldreich, O. (2001), *Foundations of Cryptography*, Vol. 1, Cambridge University Press.
- Goldreich, O. (2002), ‘Zero-knowledge twenty years after its invention’, *Electronic Colloquium on Computational Complexity* **63**.
- Goldreich, O. (2004), *Foundations of Cryptography*, Vol. 2, Cambridge University Press.
- Goldreich, O., Micali, S. & Wigderson, A. (1991), ‘Proofs that yield nothing but their validity or’, *Journal of the ACM* **38**(1), 691–729.
- Halpern, J. (2007), ‘Computer science and game theory: A brief survey’.
- Jakobsson, M. (1995), Ripping coins for a fair exchange, in L. C. Guillou & J. J. Quisquater, eds, ‘Advances in Cryptology - Proc. EUROCRYPT ’95: International Conference on the Theory and Application of Cryptographic Techniques; Lecture Notes in Computer Science Volume 921’, Springer.
- Koblitz, N. I. (1994), *A Course in Number Theory and Cryptography*, Springer Verlag.
- Mao, W. (1998), Guaranteed correct sharing of integer factorization with off-line shareholders, in ‘Proc. Public Key Cryptography, Lecture Notes in Computer Science 1431’, Springer.
- Menezes, A., van Oorschot, P. & Vanstone, S. (1997), *Handbook of Applied Cryptography*, CRC Press.
- NewScientist (2005), ‘Sea slugs solve the battle of the sexes’, *New Scientist* **188**(2521), 19–19.
- Pedersen, T. (1991), ‘Non-interactive and information-theoretic secure verifiable secret sharing’.
- Pitchford, R. & Snyder, C. M. (2004), ‘A solution to the hold-up problem involving gradual investment’, *Journal of Economic Theory* **114**, 88–103.
- Sandholm, T. (1996), ‘Negotiation among self-interested computationally limited agents’.
- Sandholm, T. (1997), ‘Unenforced e-commerce transactions’, pp. 47–54.
- Schoenmakers, B. (2005), ‘Interval proofs revisited’.
- Syverson (1998), Weakly secret bit commitment: Applications to lotteries and fair exchange, in ‘Proceedings of The 11th Computer Security Foundations Workshop’, IEEE Computer Society Press.
- Teague, V. (Forthcoming), ‘Problems in coordination with two player games’, *Econometrica* .