

# How to Safely Close a Discussion

Gildas Avoine<sup>a</sup> and Serge Vaudenay<sup>b</sup>

<sup>a</sup>*MIT, Cambridge, USA*

<sup>b</sup>*EPFL, Lausanne, Switzerland*

---

## Abstract

In the secure communication problem, we focus on safe termination. In applications such as electronic transactions, we want each party to be ensured that both sides agree on the same state: success or failure. This problem is equivalent to the well known coordinated attack problem. Solutions exist. They however concentrate on the probability of disagreement, and attack incentives have been overlooked so far. Furthermore, they focus on a notion of round and are not optimal in terms of communication complexity.

To solve the safe termination problem, we revisit the Keep-in-Touch protocol that we introduced in 2003. Considering the communication complexity, the probability of unsafe termination, and the attack incentive, we prove that the Keep-in-Touch protocol is optimal.

*Key words:* Safe Termination, Synchronization, Secure Communication.

---

## 1 Synchronization Protocol

For applications requiring secure communication, we often use standard cryptography to achieve security at the packet level in terms of *authentication*, *integrity*, and *confidentiality*. Assuming cryptography performs a good job, adversaries can still try to remove, replay, or permute packets. Standard solutions consist in authenticating a sequence number for each packet so that the *packet sequentiality* can no longer be corrupted. One remaining problem though: the adversary can still maliciously remove final packets by disconnecting the channel, depending on what she has seen so far (namely, the volume and directions of packets). To solve this, both parties must agree that the communication is over by means of a *synchronization protocol*.

In contract signing protocols, after participant Alice sent the signed contract to Bob, Bob signed it and sent it back to Alice, Bob may assume that the

contract is signed and will be executed but Alice may have never received the signature and may be assuming that Bob declined the contract. Informally, contract signing should be followed by a synchronization protocol: each party decides whether the protocol is assumed to have succeeded and launch a synchronization protocol with this input bit. At the end of the protocol, their output tells whether it is indeed agreed on a success or not.

**Definition 1** *A synchronization protocol specifies two probabilistic algorithms  $A$  and  $B$  that communicate together. Both algorithms start with an input bit and terminate with an output bit. We require that*

- $A$  and  $B$  eventually halt;
- no algorithm yield 1 if its input is 0;
- when the channel is not disconnected,  $A$  and  $B$  always yield the product of the two input bits.

*The synchronization succeeds if both  $A$  and  $B$  terminate with the same output. Adversaries are assumed to know the algorithms  $A$  and  $B$ , their input bits (but not their input coins), and can see at every step if a protocol message is emitted (but not its actual content) by either  $A$  or  $B$ . The only possible action by the adversary is to cut the communication link. The protocol quality is measured by*

- $C$ , the communication complexity in terms of number of messages that  $A$  and  $B$  exchange;
- $P_a$  (probability of asymmetric termination), the maximum of the probability that the protocol fails, over all possible adversaries;
- $P_c$  (probability that crime pays off), the maximum of the conditional probability that the protocol fails, conditioned on the protocol to be interrupted by a malicious disconnection, over all possible adversaries.

Note that there is a tricky distinction between  $P_a$  and  $P_c$  which will be shown in the sequel.  $P_a$  gives confidence to  $A$  and  $B$  that the protocol will succeed while  $P_c$  measures the incentive for misbehavior.

**Related work.** In 1978, Gray [5] introduced the Generals paradox, later called *coordinated attack problem*. In this problem, two or several participants must agree on a binary decision (attack or not) through a communication channel that can maliciously remove messages. Even and Jacobi [3] have shown that it cannot be solved with 100% probability with finite complexity so we must consider a *probability of disagreement*. As detailed below, optimal solutions have been proposed (first in 1992) in the multiparty case by Lynch and Varghese [6,8]. Problems that are related to coordinated attack include *consensus* [4], *non-blocking atomic commitment* [2,7], and *fair exchange*. Those

usually consider the Byzantine failure model, which allows arbitrary behavior of a bounded number of participants. Here, we assume that participants are all correct and that cryptography already protects the communication link except disconnection. In 2003, we introduced in [1] the notion of *Keep-In-Touch protocol* (KiT) to solve a specific fair exchange problem: two malicious participants assisted by one *guardian angel* each who try to fairly exchange digital information. It is assumed that the two guardian angels trust each other but that their communication channel go through the untrusted participants. The KiT-based solution consists of making the exchange at the guardian angels level, then making a synchronization to check whether the exchange was fair, and finally releasing the obtained digital information.

**Varghese-Lynch protocol.** In the protocol by Varghese and Lynch [6,8], the participants run a given (public) number of *rounds*  $r$  in any case (namely, even when they stop receiving messages by others). Rounds are scheduled periods of time during which the participants must exchange a message in all directions. The protocol is optimal in terms of number of rounds. (For two participants, the message complexity is indeed  $C = 2r$  and the running time is  $r$  “units” where a unit upper bounds the required time to exchange two messages. So, some time is wasted if transmission is faster than a unit.) At the beginning, the originator of the protocol picks a random number  $N \in \{1, \dots, r\}$  with uniform distribution and sends it to his counterpart. Each party manages a counter initialized to 0, whose value is sent in each message. At the end of a round, the internal counter is synchronized with the received one (if any). At the beginning of a round, it is incremented by 1. The secret number  $N$  picked by the originator is a threshold used at the end of the exchanges: each party outputs 1 if and only if its counter is at least equal to  $N$ . So, the only way for the adversary to injure the parties is to guess the value  $N$ , and then to cut any messages, provided that one counter is equal to  $N$  while the other is smaller. Analysis shows that  $P_a = P_c = \frac{1}{r}$  ([6,8] only consider  $P_a$  but we can easily show that  $P_c$  is the same in the case of this protocol).

**Our contribution.** In this paper, we focus on average communication complexity in terms of number of exchanged messages. We distinguish the measures  $P_a$  and  $P_c$  (while previous work concentrated on  $P_a$ ). Then, we formalize a variant of the Keep-In-Touch protocol (KiT). We show that it is a synchronization protocol. Finally, we prove optimality in terms of number of exchanged messages. It is notably more efficient than the Lynch-Varghese protocol with two participants since the running time no longer depends on synchronized rounds and that we achieve  $P_a = P_c \leq \frac{1}{E(C)-1}$  instead of  $P_a = P_c = \frac{2}{C}$ .

## 2 Keep-in-Touch Protocol

We propose a variant of the KiT protocol (that we call KiT protocol as well in the sequel) as depicted on Fig. 1. The principle is quite simple: if the input of  $A$  is 1 then  $A$  picks a random number  $N \geq 0$  and sends it to  $B$  (note that the confidentiality of  $N$  is protected).  $N$  represents the number of messages that should be exchanged after  $B$  joined the protocol by sending his first message. Depending on the probability distribution for  $N$ , it can be bounded or not. Then, if both inputs are 1,  $A$  and  $B$  just keep in touch by sequentially exchanging authenticated messages. Contrarily to the first message which includes  $N$ , the  $N + 1$  other messages are really empty ones! In case of time-out while expecting a message, a participant stops and yields 0.

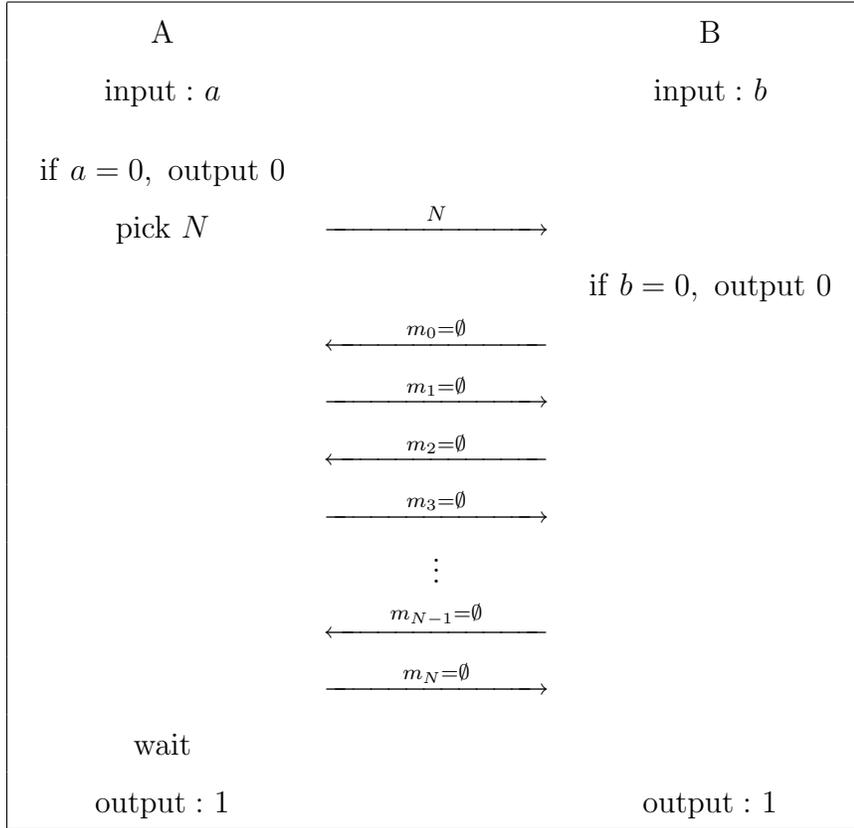


Fig. 1. Keep-in-Touch (KiT) Protocol

**Termination side channel protection.** In the case where the adversary has access to the output of  $A$  or  $B$  through a side channel, the last sender should wait for a given period larger than the time-out before terminating. This makes sure that both  $A$  and  $B$  complete before the adversary gets any side information. The last receiver might still acknowledge for the last message to prevent the other party from waiting, but disconnection at this point should

not change the output. The consequence of such an attack is only a time loss for the waiting participant.

**Time-out removal.** Similarly, when  $a = 0$ ,  $A$  can prevent  $B$  from waiting by sending a specific message. The  $a = 1$  and  $b = 0$  case is similar.

**Theorem 1** *The KiT Protocol is a synchronization protocol. Let  $p_0, p_1, \dots$  denote the probability distribution of  $N$  in the protocol, i.e.,  $p_i = \Pr[N = i]$ . The average message complexity is such that  $E(C) \leq 2 + E(N)$  where  $E(N) = \sum_i ip_i$ , the probability of asymmetric termination is  $P_a = \max_i p_i$  and the probability that the crime pays off is  $P_c = \max_i (p_i / \sum_{j \geq i} p_j)$ .*

**Proof 1** *When no attack occurs and  $a = b = 1$ , the complexity in terms of exchanged messages is exactly equal to  $C = N + 2$ . When the channel is cut or  $ab = 0$ , the complexity is smaller, so we can just focus on  $N$ . By definition, the average complexity is  $2 + E(N)$ , where  $E(N) = \sum_i ip_i$ . Note that the communication and time complexities are linear in terms of  $N$  due to the simplicity of the message contents and the computations to perform.*

*Obviously, all properties in Definition 1 are satisfied so we have a synchronization protocol. We now measure its quality in terms of  $P_a$  and  $P_c$ .*

*$P_a$  computation. Clearly, disconnecting  $A$  from  $B$  in the first message makes  $A$  and  $B$  output 0 and the protocol succeeds. We now assume that the adversary is willing to drop  $m_i$ . If  $N < i$  then the attack has no influence and the protocol successfully terminates. If  $N > i$ , the participant who is expecting  $m_i$  cannot send the next one, so both participants are blocked and the protocol succeeds since both  $A$  and  $B$  yield 0 after time-outs expire. Clearly, the protocol fails only if  $N = i$ , thus with probability  $p_i$ . Therefore we have  $P_a = \max_i p_i$ .*

*$P_c$  computation. With the same discussion we can show that the above misbehavior has a conditional probability of success of  $\Pr[N = i | N \geq i]$ . Hence we have  $P_c = \max_i \frac{p_i}{\sum_{j \geq i} p_j}$ .  $\square$*

**Example 1** *For any  $n$ , when  $p_0 = \dots = p_{n-1} = \frac{1}{n}$  and  $p_i = 0$  for  $i \geq n$  we have  $E(N) = \frac{n-1}{2}$  thus  $E(C) \leq \frac{n+3}{2}$  and a probability of asymmetric termination of  $P_a = \frac{1}{n}$ . The worst case complexity is  $C = n + 1$ . However we have  $P_c = 1$  (with  $i = n - 1$ ). In other words, the longer the adversary waits before performing his attack, the greater the probability the attack succeeds, in particular if his strategy is to disconnect at  $m_{n-1}$  then his attack definitely succeeds.*

**Example 2** *For any  $p$ , when  $p_i = (1 - p)^i p$  for  $i \geq 0$  we have  $E(N) = \frac{1}{p} - 1$  thus  $E(C) \leq \frac{1}{p} + 1$  and a probability of asymmetric termination of*

$P_a = p$ . In this case we also have  $P_c = p$ . With the same  $P_a$  and  $P_c$  as in the Lynch-Varghese protocol, our protocol has an average complexity at most  $r + 1$  messages instead of  $2r$ . It does not waste time. However, the worst case complexity is not bounded.

**Optimal distributions for the KiT protocol.** The distribution choice plays on the message complexity and the parameters  $P_a$  and  $P_c$ . Obviously there is a trade-off. The optimal case is studied in the following theorem.

**Theorem 2** *Let  $p_0, p_1, \dots$  denote the probability distribution of  $N$  in the KiT protocol. We have  $E(N) \geq \frac{1}{2} \left( \frac{1}{P_a} - 1 \right)$  and  $E(N) \geq \frac{1}{P_c} - 1$  where  $P_a$  and  $P_c$  are the probability of asymmetric termination and the probability that the crime pays off respectively.*

This shows that Example 1 is the optimal case for  $P_a$  and that Example 2 is the optimal case for  $P_c$ .

**Proof 2** *We want to minimize  $E(N)$  for a given  $P_a$ . Due to Theorem 1, it is equivalent to finding  $p_0, p_1, \dots$  such that  $0 \leq p_i \leq P_a$  for all  $i$ ,  $\sum p_i = 1$ , and  $\sum ip_i$  minimal.*

Let  $n = \lfloor \frac{1}{P_a} \rfloor$  and  $\alpha = \frac{1}{P_a} - n$ . We have  $\alpha \in [0, 1[$ .

Obviously  $\sum ip_i$  is minimal when the first  $p_i$ s are maximal, i.e., when  $p_0 = p_1 = \dots = p_{n-1} = P_a$ . The sum of all remaining  $p_i$  is equal to  $1 - nP_a$ . Thus we have

$$E(N) \geq P_a + 2P_a + \dots + (n-1)P_a + n(1 - nP_a).$$

Hence  $E(N) \geq \frac{n(n-1)}{2}P_a + n(1 - nP_a)$ . If we substitute  $\frac{1}{P_a} - \alpha$  to  $n$  we obtain

$$E(N) \geq \frac{1}{2} \left( \frac{1}{P_a} - 1 \right) + \frac{\alpha P_a}{2} (1 - \alpha).$$

Since  $0 \leq \alpha < 1$  we have  $E(N) \geq \frac{1}{2} \left( \frac{1}{P_a} - 1 \right)$ . This proves the first bound.

For the second bound we notice that

$$E(N) = \sum_{i=1}^{+\infty} \sum_{j \geq i} p_j = \sum_{i=0}^{+\infty} \sum_{j \geq i} p_j - 1.$$

Since we have  $\sum_{j \geq i} p_j \geq \frac{p_i}{P_c}$  for all  $i$  due to Theorem 1, we obtain that  $E(N) \geq \frac{1}{P_c} - 1$ .  $\square$

**Bit-messages variant.** Instead of picking  $N$  once and sending it at the beginning of the protocol, we can just ask each participant to toss a coin before sending  $m_i$  and sending the result of the toss in the message. “Head” means “let’s keep in touch” and “tail” means “so long”. Obviously, if the coin is biased such that the  $i$ th toss is “tail” with probability  $\Pr[N = i | N \geq i]$ , this variant is fully equivalent to the above protocol. Example 2 is equivalent to the case where the probability to get “tail” is  $p$  for all  $i$ .

### 3 Optimality of the KiT Protocol

We prove in this section that our protocol is optimal within our settings. This also (re)proves that perfect synchronization cannot be ensured with  $P_a = 0$  or  $P_c = 0$  with a finite message complexity (since KiT protocols do not).

**Theorem 3** *For any synchronization protocol between two participants  $A$  and  $B$  (initiated by  $A$ ) with parameters  $P_a$  and  $P_c$ , we let  $C = N + 2$  denote the number of exchanged messages. We can define a random variable  $N'$  such that  $\Pr[N' \leq N] = 1$  and that  $N'$  defines a KiT protocol  $A'B'$  with parameters  $P'_a$  and  $P'_c$  such that  $P_a \geq P'_a$  and  $P_c \geq P'_c$ .*

**Proof 3** *We are given two (probabilistic) algorithms  $A$  and  $B$  which exchange  $N + 2$  messages in normal conditions. Without loss of generality, we assume that  $A$  initiates the protocol. We now construct a KiT protocol  $A'B'$  which uses  $N' + 2$  messages. Note that we only need to define how  $A'$  computes  $N'$  since the remaining part of  $A'$  and  $B'$  are fully specified by the KiT protocol.*

*We first note that when either input is 0, the KiT protocol is always optimal: sending less messages may lead to cases which would violate the definition of the synchronization protocol. We deduce that  $N$  must be positive when both inputs are 1. We concentrate in this case in what follows.*

*In order to compute  $N'$ ,  $A'$  first simulates a normal interaction between  $A$  and  $B$  with input 1. We assume that all random coins are set in advance by  $A'$  so that the simulation is run on deterministic algorithms. Note that the simulator can freely restart  $A$  or  $B$  in a previous state. Hence,  $A'$  can define the following quantities based on a random instance of the simulation. Let  $x_1, x_2, \dots, x_{N+2}$  be a sequence of bits in which  $x_i$  is equal to 0 when the  $i$ th message is sent from  $A$  to  $B$ , and to 1 when it is sent in the other direction. By definition of the synchronization protocol, both  $A$  and  $B$  yield 1 after the final message.  $A'$  now analyzes the final output in case the adversary disconnects the channel at the  $i$ th message (i.e., this message is sent but never received) for  $i = 1, \dots, N + 2$ . We define  $a_i$  (resp.  $b_i$ ) as the final output of  $A$  (resp.  $B$ ) if the channel is disconnected at the  $i$ th message. We let  $N' + 2$  be the smallest  $i$  such that*

$a_j = b_j = 1$  for any  $j \geq i$ . Obviously we always have  $N' \leq N$ . In the rest of the proof we demonstrate that the  $P'_a$  and  $P'_c$  parameters for the  $A'B'$  protocol are no larger than the  $P_a$  and  $P_c$  parameters for the  $AB$  protocol. In order to do this we show that any attack strategy  $S'$  against  $A'B'$  can be transformed into an attack strategy  $S$  against  $AB$  with at least the same probability of success.

An attack  $S'$  against  $A'B'$  is fully defined by the index  $i$  of the message from which the channel is disconnected. We consider the attack  $S$  against  $AB$  which cuts the channel when the  $i$ th message is sent. The attack  $S'$  succeeds only for instances of the  $AB$  simulation in which  $i = N' + 2$ . We show below that  $S$  also succeeds for the same instances. We deduce that no attack against  $A'B'$  is more successful than any attack against  $AB$ . Hence  $P_a \geq P'_a$  and  $P_c \geq P'_c$ .

Let us assume that  $x_i = 0$ . By definition of  $N'$  we have  $a_{N'+2} = b_{N'+2} = 1$ , but we do not have  $a_{N'+1} = b_{N'+1} = 1$ . We notice that  $a_{N'+1} = a_{N'+2}$  since everything is normal for  $A$ . Thus we have  $b_{N'+1} = 0$ . Since  $B$  does not receive the  $i$ th message,  $B$  eventually yields 0 while  $A$  yields 1. A similar argument holds for  $x_i = 1$ .  $\square$

## Acknowledgments

The authors would like to thank Moti Yung and André Schiper for having provided pertinent references and helpful comments on this work.

## References

- [1] G. Avoine, S. Vaudenay, Fair exchange with guardian angels, in: K. Chae, M. Yung (Eds.), International Workshop on Information Security Applications – WISA 2003, Vol. 2908 of Lecture Notes in Computer Science, Springer-Verlag, Jeju Island, Korea, 2003, pp. 188–202.
- [2] P. Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley Publishers, Boston, Massachusetts, USA, 1987.
- [3] S. Even, Y. Yacobi, Relations among public key signature systems, Technical Report 175, Computer Science Department, Technion, Haifa, Israel (1980).
- [4] M. Fischer, N. Lynch, M. Paterson, Impossibility of distributed consensus with one faulty process, Journal of the Association for Computing Machinery 32 (2) (1985), pp. 374–382.

- [5] J. Gray, Notes on data base operating systems, in: R. Bayer, R. Graham, G. Seegmüller (Eds.), *Operating Systems, An Advanced Course*, Vol. 60 of *Lecture Notes in Computer Science*, Springer-Verlag, 1978, pp. 393–481.
- [6] N. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, San Francisco, California, USA, 1996.
- [7] D. Skeen, Non-blocking commit protocols, in: E. Lien (Ed), *ACM SIGMOD International Conference on Management of data*, ACM, ACM Press, Ann Arbor, Michigan, USA, 1981, pp. 133–142.
- [8] G. Varghese, N. Lynch, A tradeoff between safety and liveness for randomized coordinated attack, *Information and Computation* 128 (1) (1996) 57–71.