The Spymasters Double-Agent Problem
Multiparty Computation Secure Unconditionally from Minorities and Cryptographically from Majorities

David Chaum

Centre for Mathematics and Computer Science
Kruislaan 413 1098 SJ Amsterdam

SUMMARY

A multiparty-computation protocol allows each of a set of participants to provide secret input to a mutually agreed computation. Such protocols enforce two security properties: (1) secrecy of the inputs, apart from what is revealed by the output; and (2) correctness of the output, as defined by the agreed computation. All solutions, including those presented here, are based on two kinds of assumptions: (a) public-key cryptography; and (b) limited collusion in a setting where pairs of participants can exchange messages with secret and authenticated content. Some of the previous solutions relied totally on assumption (a), the others totally on (b).

The main result presented here is a protocol that also provides both security properties, (1) and (2), but that does not rely on either assumption (a) or assumption (b) alone—security can be violated only by violating *both* assumptions.

The second construction improves the previously published multiparty computation results based on assumption (b). Let the number of participants be n, the largest tolerable number of disrupters be d, and the largest tolerable number of participants in any collusion be c. (Note that many collusions may exist, even to the extent that all participants are involved, but c is the maximum number of participants in any single collusion.) The construction requires $n > 2d + c$ and $n > 2c$. The first inequality gives a trade-off between the number of disrupters and the largest collusion size, which includes the previously achieved case of both less than a third. The second inequality, which means that all collusions of minorities can be tolerated, is argued to be optimal and makes the main result also optimal.

G. Brassard (Ed.): Advances in Cryptology - CRYPTO '89, LNCS 435, pp. 591-602, 1990.

A third construction, on which the second is based but which is interesting in its own right, is that of an "all-honest world." This is a setting, relying only on assumption (b), in which any participant who has revealed secrets to any other can prove publicly that the secrets revealed are correct and receivable by the second participant—even if the second participant denies receipt or correctness.

1 INFORMAL INTRODUCTION

A spymaster's deepest fear, it might be said, is that of a "double agent." If the spymasters of major countries would be willing to pool all the information they have on their agents, then they could discover—to their mutual benefit— all double agents who play one side off against the other. But for a spymaster, revealing this sensitive data to "the other side" is, of course, unthinkable.

A solution to the spymasters' problem illustrates the main result achieved here: optimal security for general multiparty computations, given only cryptography and diplomatic pouches. And since these are the means available to spymasters, this is the kind of security they require.

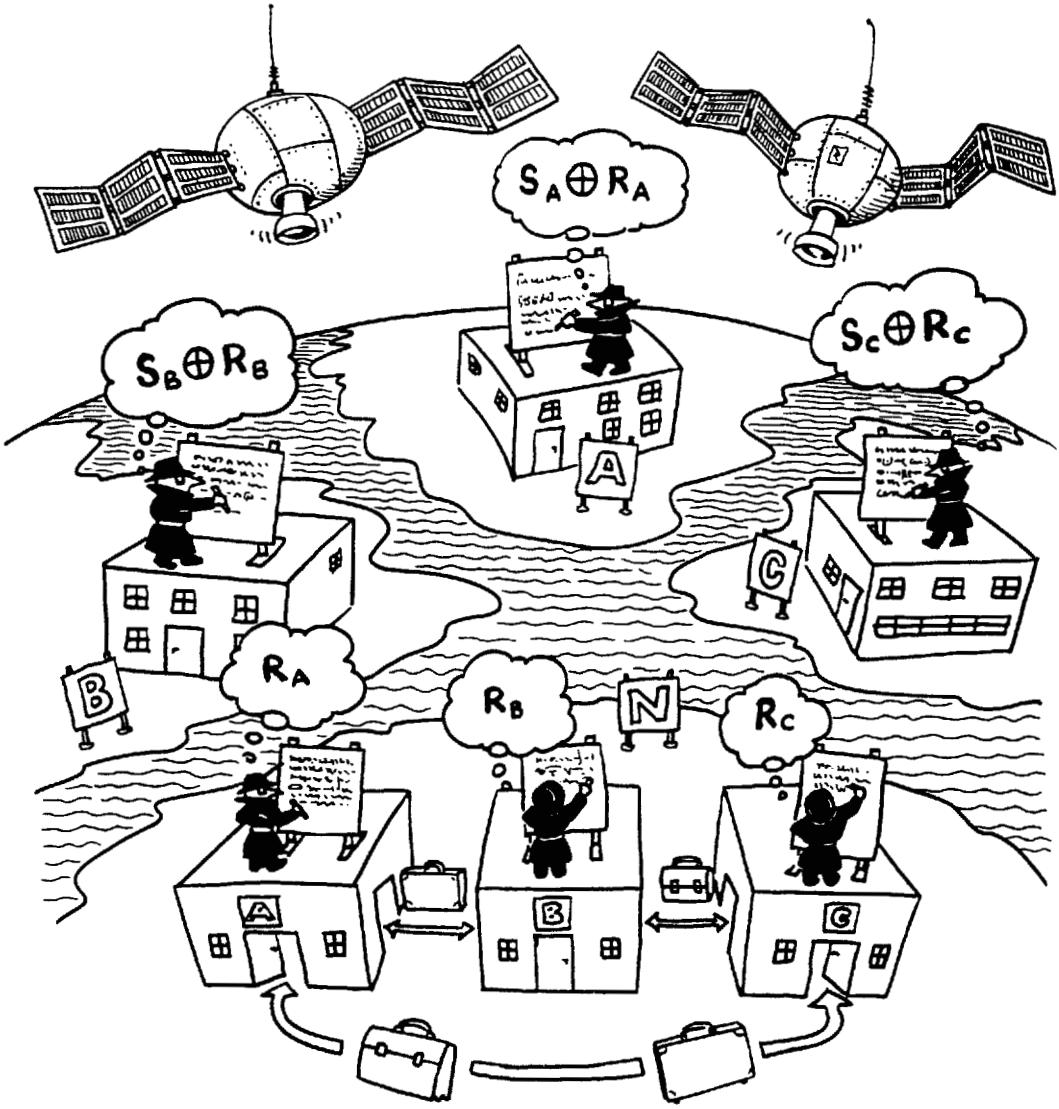
If only the spymasters could use a physical computer that they all trust. Then they could simply supply their ultra-secret dossiers on each agent as input to a mutually agreed program that would derive and output the identities of all double agents. It is assumed that a suitable program can be agreed on. The only difficulty is the computer: How could such a device be physically built and operated securely? (But see [C].)

Spymasters know, from the literature, that the effect of such a mutually trusted computer can be achieved merely by exchanging messages. They know also that two quite different kinds of protocols have been proposed for this. The most recent type [CCD & BGW] requires only that each pair of participants exchange messages in a way that ensures authenticity and secrecy of message content. This the spymasters can readily achieve by diplomatic pouch and courier. The problem they have with this kind of approach, however, is that if a sufficient number of countries collude, these countries can learn all the secret-agent profiles of the other countries.

The earlier kind of protocol in the literature [GMW2 & CDG] does not have this problem; with it, collusion yields no advantage. Its drawback, though, is that secrecy relies on public-key cryptography. Thus, if some country were able to break the agreed public-key system, its intelligence service could clandestinely learn all the other countries' secrets.

Neither approach alone is optimal and hence acceptable to the spymasters. And simply conducting both kinds of protocol in parallel would be ridiculous, since it would give the disadvantages of both—a country breaking the cryptosystem could discover all other countries' secrets, and any sufficient collusion could also learn the secrets.

The new techniques presented here allow the best of both approaches in a single protocol. No collusion of countries is sufficient to obtain secrets of non-colluders; nor does breaking the cryptosystem yield any information whatsoever. The only way some countries can learn the secrets of others is for a collusion of a majority of countries to break the cryptosystem.



1.1 EXAMPLE OF THE CONSTRUCTION

The figure shows a setting with three countries a, b, c. In addition to national headquarters buildings, the countries have embassies located near one another in a neutral zone N. The embassies' mutual proximity is convenient, since couriers will transfer pouches containing secret messages between them. Headquarters use only another means of communication, which is also used by the embassies: staff members who write messages on rooftop blackboards. All countries are ensured of obtaining the identical message "broadcast" in this way, via their spy satellites.

The spymaster of each country a, b, c has secret input for the computation S_a, S_b, S_c, respectively. A spymaster (not shown) does not provide these most sensitive secrets to the embassy or headquarters staff (shown on the rooftops). Instead, spymaster i "Feistels" S_i into two parts [F], a random string R_i and the bit-wise exclusive-or sum $R_i \oplus S_i$, and personally delivers the first part to the embassy and the second to headquarters. Notice that this arrangement means, for example, that what is known to the headquarters of country b alone, $S_b \oplus R_b$, reveals nothing about S_b; similarly, what is known to b's embassy, R_b, also gives no clue about S_b.

1.2 THE PROTOCOL

To uncover double-agents, both types of protocols for multiparty computations are used—but in a special way. The cryptographic type is performed as a four-party protocol. Each headquarters is a party to the protocol, and the fourth party is played by the embassies outputting in unison. This means that the embassies, whenever they are required to do so by the four-party protocol, must all write the same thing on their blackboards.

To decide what to write, the embassies together perform a three-party protocol. They do this every time they must write something for the four-party protocol; thus, they perform one complete three-party protocol each time the four-party protocol requires a contribution from them. (Consistency across three-party protocols is ensured by "bit commitments.") These three-party protocols use pouches to provide security that does not depend on cryptography. (To achieve optimal security, as detailed later, they also use the blackboards, but only while the four-party protocol awaits their decision.)

The embassies provide outputs to the four-party protocol that are the same as would be provided by a single party knowing all of the R_i's—but the three-party protocols prevent any embassy from learning more than its R_i. The function

f computed by the four-party protocol is $f(R_a||R_b||R_c, R_a \oplus S_a, R_b \oplus S_b, R_c \oplus S_c) = e(S_a, S_b, S_c)$, where e is the agreed double-agent outputing function and "||" denotes concatenation. Thus, the computation of f by the four-party protocol first X-ORs out the R_i's (that it gets as inputs from the embassies) from the headquarters inputs, and then computes e on the S_i.

1.3 WHY IS IT SECURE?

The cryptographic protocols of [CDG] offer optimal security, in the sense that they allow a single designated participant whose secret input is protected without any reliance on cryptographic assumptions. This participant is played by consensus of the embassies, which is itself a protocol that also does not require cryptographic assumptions. Thus, the $S_b \oplus R_b$, which reveal nothing about the S_i, are the only inputs exposed to cryptanalysis. On the other hand, the only protocol vulnerable to collusion is the embassy consensus protocol; but its only inputs are the R_i, which also reveal nothing about the S_i.

2. COMBINING CRYPTOGRAPHY AND POUCHES

This section treats the protocol introduced in the previous section more precisely. First it makes the model explicit. Then it describes the protocol, relying on the introduction of the previous section. Finally, the main result is contained in two theorems: one for secrecy of the inputs; the other for a topic ignored in the previous section, correctness of the output.

2.1 MODEL

The construction is based on two assumptions:

(a) Trap-door one-way bijections and "claw-free" functions exist. (Such assumptions underlie the cryptographic protocol [CDG] and are satisfied by the well-known quadratic residuosity assumption [GM].)

(b) Less than the specified number of participants collude and each pair of participants can communicate with secrecy and authentication. (This assumption underlies the protocols of [CCD] and [BGW]. The channel required can be achieved in practice in various ways: by exchanging long keys in person and then using a one-time pad and corresponding authentication coding; by exchanging short keys in person and then using a conventional cryptosystem; or perhaps even by realizing quantum cryptography [BB].)

2.2 PROTOCOL

The participants in the protocol correspond to the spymasters, of whom there are n. This means that the i'th participant, $1 \le i \le n$, knows both S_i and R_i. Thus, the participant is involved in one n+1-party [CDG] protocol and also in a number of n-party protocols of the type presented in [CCD] or [BGW] or section 3.

The computations performed by the protocols are as described in section 1. A technical point only hinted at there, though, is the source of randomness used by the n-party protocols. The X-OR of unconditionally-privacy-protecting bit commitments issued initially by all participants can be used as the "random tape" of the computation playing the role of the n+1'th participant. Participants then show, by "blob equality" [BCC], that their input to each protocol round is consistent with their contributions to the random tape.

2.3 PROPERTIES

The protocol has the following two security properties:

Theorem 2.1: The secrecy of each participant's input is protected unless both assumptions (a) and (b) are violated.

Proof: (Sketch) The inputs to either individual protocol by a participant following protocol are statistically independent of that participant's secrets. The output of the protocol that relies on assumption (b) only enters the other protocol through a predetermined participant, whose privacy also depends only on (b). Thus, it is necessary (and sufficient) to violate both (a) and (b) in order to gain information about a participant's secrets. Q.E.D.

Theorem 2.2: The correctness of the output is ensured with probability exponentially high in a security parameter unless both (a) and (b) are violated.

Proof: (Sketch) The [CDG] protocol gives exponential certainty that the first n participants cannot cause incorrect output. The n+1'th participant can falsify output, but only by violating assumption (a). For the pouch-based protocols playing this n+1'th participant, the correctness of their contribution is guaranteed with exponential certainty unless assumption (b) is violated ([BGW] achieve a stronger result of not allow even an exponentially small chance of cheating). Thus, violation of both (a) and (b) is necessary to give a non-negligible probability of false output. Q.E.D.

3 IMPROVED POUCH CONSTRUCTION

The model underlying this section is stated in §3.1. Next, §3.2 describes a "cut-and-choose" originally proposed by [Be] for other purposes, and used by [CCD] and [BGW]. Then §3.3 presents the "all-honest world" construction. Finally, §3.4 shows how circuits are simulated, using the essence of the "double-degree polynomial" trick proposed in both [CCD] and [BGW].

3.1 MODEL

As already mentioned, the number of participants is denoted n, the largest tolerable number of disrupters d, and the largest tolerable number of participants in any single collusion c.

Disrupters are defined as participants whose outputs do not follow protocol. Once a participant is agreed to be a disrupter, the protocol can (if necessary) be restarted without that participant (but see §4); this is why disrupters may try to falsely blame others for sending them improper messages. Violation of property (2), correctness of the result, requires active cheating, and hence disrupters.

A collusion, on the other hand, is a set of participants who merely share their information in efforts to learn the secret input of others. Because a collusion could even be a secretly conducted instance of the type of protocol described here, each participant could be a member of multiple collusions. It will be sufficient, however, simply to ensure that no collusion has access to information from more than c participants.

The construction requires $n > 2d+c$ and $n > 2c$. If $n = 3$, for instance, then $d = 0$, which means that even a single disrupter can falsify the output; but, since even this small n allows $c = 1$, secrecy of the inputs can be protected unconditionally against any participant acting alone. When $n = 4$, a single disrupter can be tolerated. More generally, $c = 1$ allows almost half the participants to be disrupters. At the other end of the trade-off, with d at about a quarter n, any c less than half n is possible. This last is optimal; otherwise, disjoint sets of participants could conduct an arbitrary two-party protocol with both parties protected unconditionally—and this, as argued in [CDG], is impossible.

3.2 SET-UP BLOBS

Included in the mutually agreed and public protocol definition are: an integer k such that $2^k > n$; an assignment of a distinct point in GF(2^k) to each participant; and a security parameter s.

A participant issues a *blob* by first choosing a polynomial of degree at most c, uniformly over GF(2^k), with a value at 0 of 1 or 0. Then the issuing participant uses the corresponding pouch to supply every other participant with a *share*—the value of the polynomial at that other participant's point. When a blob must be *opened*, every participant broadcasts the share it holds for the blob and the issuer separately broadcasts what each other participant should output.

Every blob used in the remaining protocol is subjected to s challenges by each participant. Consider a single challenge of a particular participant. First, the issuer creates a new blob, and then the challenger broadcasts a random bit. If this challenge bit is 0, the new blob is opened (as defined above); if it is 1, each participant computes the sum of the share it held of the original blob with the share of the new blob, and the resulting sum blob is opened.

If the challenger and issuer disagree on the value of the polynomial at the challenger's point, the challenger is said to *object*; further objections can result if and when the original blob is opened. If more than d participants ever object for any single issuer, then that issuer is clearly indicated as a disrupter (under the assumption of at most d disrupters) and the protocol terminates.

Theorem 3.1: If the number of objections for a blob does not exceed d, then it can be opened both as 1 and 0 with probability at most 2^{-s}.

Proof: (Sketch) With probability $1-2^{-s}$, all non-objecting non-disrupters, of whom there are at least $c+1$, will broadcast shares consistent with a single polynomial p. This is a simple consequence of the challenge and response technique. Because each polynomial has degree at most c, it is always completely determined by $c+1$ shares. Since $d+c+1$ consistent shares are broadcast during opening, at least d shares may be called redundant because they are consistent with p but are not necessary to determine p. For the blob to be opened as both 1 and 0, shares would have to be consistent with two distinct polynomials, p and q. If $d+c+1$ shares are consistent with p, then $d+1$ must be changed to be consistent with q, since the redundancy means that changing any d or fewer shares leaves sufficient shares to determine p. Thus at least $d+1$ shares must be changed, so $d+1$ disrupters are implied. Q.E.D.

3.3 THE ALL-HONEST WORLD

During the protocol proper, pouches will not be used (except possibly for Byzantine agreement [LPS] of broadcasts when $n \leq 3d$). Instead, by setting up an "all-honest world," the participants arrange in advance for every bit of message

that will be sent between them. For each such message bit to be sent in the all-honest world, the parity of the bits in two blobs—one blob issued by each communicant—is made public in advance. There are two cases.

In the first case, the cardinality of the union of the objector sets for the two communicants does not exceed d. The intersection of the non-objecting sets for the two communicants thus contains at least $d+c+1$ participants. Two blobs are issued, one by each of the two communicants, and the sum of the two blob is opened. Such "opening" differs from that for an ordinary blob, since the issuers will not broadcast the shares they issued. If the shares broadcast by the participants in the intersecting set are consistent with a single polynomial having a binary value at zero, then this value at zero is the public parity bit. If no such bit is recognizable, both blobs are opened separately (which adds at least one member to one of the two objector sets) and the process is repeated for a new pair of blobs.

In the second case, that cardinality of the union of objector sets exceeds d, there will be $c+1$ participants, called a *common* set, who will successfully satisfy the first case with each of the pair of communicants. (To see this, notice that the second case implies that one of the two communicants is a disrupter—thus, a participant involved in such situations with d other participants is recognizable as a disrupter.) Consider, without loss of generality, a particular pair of communicants with a common set and a particular participant in that common set. This common participant will use two blobs satisfying the first case, one with each of the two communicants. The common participant asks everyone to add their shares corresponding to the two blobs, and opens this sum blob to reveal the parity of the contents of the two original blobs. The channel parity bit is then easily computed as an X-OR of all such bits made public by the common set.

To send a bit in the all-honest world once all the parity bits have been established, the sender simply makes public the actual message bit X-ORed with the sender's contribution to the parity bit. Then the sender can prove the correctness of the bit sent by using the sender's other blobs in a general satisfiability protocol, like that of [BCC] or [GMW1]. The secrecy of the transmission is ensured because at least $c+1$ other participants must collude to recover the secret contents of all blobs used in establishing the parity bit. The receiver can also use the bits received in proofs related to subsequent outputs, since these bits can be expressed as sums involving only public bits and the content of the recipient's own blobs.

3.4 THE PROTOCOL PROPER

The actual secret input of participants to the computation itself will be committed to by blobs issued in the all-honest world. When a share is sent in such a blob-issuing, the sender proves that the correct value of the share has been sent, and that it is receivable, based on the public parity bits. The blobs used in this proof are all from the initial set-up phase and include blobs committing to the "random tape" that determines the choice of polynomials.

When two bits in the circuit simulated by the computation (either actual secret inputs or intermediate values) are to be X-ORed, each participant adds the share they hold for each of the bits, which yields their share for the new bit.

When two bits are to be ANDed, each participant first multiplies the two shares held, yielding a share of a "double degree" blob having maximal degree $2c$. Then the resulting share is added to a share from n new "double-degree" blobs, one issued in the all-honest world by each participant. This sum double-degree blob is then "opened" in the all-honest world; each participant opens its part of the parity bits involved. A second blob is also issued in the all-honest world by each participant; it is shown to be properly formed and to contain the same bit as the corresponding double-degree blob issued by that participant. The result of the AND-gate is then formed by each participant as the X-OR of all second blobs, or the inverse of this, depending on whether the double-degree blob opened contains 0 or 1, respectively.

4 RELEASE OF RESULTS

Issuing an actual secret input as a blob in the all-honest world costs the issuer by creating exposure to collusion. To spread such exposure evenly, participants can reciprocally commit to more and more information about their secret inputs in a "gradual commit," which is essentially the inverse of the "release of secrets" notion surveyed in [BCDG].

After the gradual commit, participants could still be robbed of the benefit of their new exposure if the computation were not completed. Any participant can stop the all-honest-world computation, but not without being recognized by the others as having done so. Even if $2d$ participants stop, however, the $c+1$ remaining ones can input all the shares they received in the all-honest world to a new protocol with a reduced n (and, consequently, possibly a reduced c). This new protocol—which has information already proven correct and sufficient to determine all the original secret inputs—computes the same result as the original computation would have.

CONCLUSION

Some earlier results are extended, improved, and generalized here; and two previously unlinked but fundamental sets of results are unified.

ACKNOWLEDGEMENTS

It is a pleasure to acknowledge all the discussion with my coauthors Claude Crépeau and Ivan Damgård on our joint work [CCD] that laid a foundation for section 3. It is also a pleasure to thank Jurjen Bos for his help in simplifying and improving the presentation.

REFERENCES

[BB] Bennett and Brassard: An update on quantum cryptography. Proc Crypto 84, pp. 474–480.

[BCC] Brassard, Chaum and Crépeau: Minimum disclosure proofs of knowledge. JCSS October 1988, pp. 156–189.

[BCDG] Brickel, Chaum, Damgård and van de Graaf: Gradual and verifiable release of a secret. Proc. Crypto 87, pp. 156–166.

[Be] Benaloh: Secret sharing homomorphisms: keeping shares of a secret secret. Proc. Crypto 86, pp. 251–260.

[BGW] Ben-Or, Goldwasser and Wigderson: Completeness theorems for non-cryptographic fault-tolerant distributed computation. Proc. STOC 88.

[C] Chaum: Computer systems established, maintained, and trusted by mutually suspicious groups. Memorandum UCB/ERL M79/10, U.C. Berkeley, February 22, 1979

[CCD] Chaum, Crépeau and Damgård: Multiparty unconditionally secure protocols. Proc. STOC 88. (To appear in JCSS.)

[CDG] Chaum, Damgård and van de Graaf: Multiparty computations ensuring privacy of each party's input and correctness of the result. Proc. Crypto 87, pp. 87–119.

[F] Feistel: Cryptographic coding for data-bank privacy. RC 2827, IBM Research, Yorktown Heights, March 1970.

[GM] Goldwasser and Micali: Probabilistic encryption. JCSS, April 1984, pp. 270–299.

[GMW1] Goldreich, Micali and Wigderson: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. Proc. of FOCS 86, pp. 174–187.

[GMW2] Goldreich, Micali and Wigderson: How to play any mental game, Proc. of STOC 87.

[GV] Goldreich and Vainish: How to play any mental game: an efficiency improvement. Proc. Crypto 87, pp. 73–86.

[LPS] Lamport, Shostak and Pease: The Byzantine generals problem. ACM trans. Prog. Languages and Systems, 1982, pp. 382–401.