RWTH Aachen

Department of Computer Science
Technical Report

Optimal Randomized Fair Exchange with Secret Shared Coins

Felix Freiling, Maurice Herlihy and Lucia Draque Penso

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2005-22

RWTH Aachen · Department of Computer Science · December 2005

Optimal Randomized Fair Exchange with Secret Shared Coins

Felix Freiling[1], Maurice Herlihy[2*], and Lucia Draque Penso[3**]

[1] Computer Science Department, University of Mannheim, D-68131 Mannheim, Germany
[2] Computer Science Department, Box 1910, Brown University, Providence, RI 02912, U.S.A.
[3] Computer Science Department, RWTH Aachen University, D-52056 Aachen, Germany

Abstract. In the *fair exchange* problem, mutually untrusting parties must securely exchange digital goods. A fair exchange protocol must ensure that no combination of cheating or failures will result in some goods being delivered but not others, and that all goods will be delivered in the absence of cheating and failures.

This paper proposes two novel randomized protocols for solving fair exchange using simple trusted units. Both protocols have an optimal expected running time, completing in a constant (3) expected number of rounds. They also have optimal resilience. The first one tolerates any number of dishonest parties, as long as one is honest, while the second one, which assumes more aggressive cheating and failures assumptions, tolerates up to a minority of dishonest parties.

The key insight is similar to the idea underlying the *code-division multiple access* (CDMA) communication protocol: outwitting an adversary is much easier if participants share a common, secret pseudo-random number generator.

1 Introduction

In the *fair exchange* problem, a set of parties want to trade an item which they have for an item of another party (for a survey of fair exchange see [11]). Fair exchange is a fundamental problem in domains with electronic business transactions since (1) items can be any type of electronic asset (electronic money, documents, music files, etc.) and (2) fairness is especially important in rather anonymous environments without means to establish mutual trust relationships. Briefly spoken, fair exchange guarantees that (1) every honest party eventually either delivers its desired item or aborts the exchange, (2) the exchange is successful if no party misbehaves and all items match their descriptions, and (3) the exchange should be fair, i.e., if the desired item of any party does not match its description, then no party can obtain any (useful) information about any other item. Fair exchange algorithms must guarantee these properties even in the presence of arbitrary (malicious) misbehavior of a subset of participants.

Fair exchange, a security problem, can be reduced [2] to a fault-tolerance problem, namely a special form of *uniform consensus*. In the (non-uniform) consensus problem [13], each process in a group starts with a private input value, and

* Maurice Herlihy was supported by Deutsche Forschungsgemeinschaft (DFG) when visiting RWTH Aachen University.

** Lucia Draque Penso was supported by Deutsche Forschungsgemeinschaft (DFG) as part of the Graduiertenkolleg "Software for Mobile Communication Systems" at RWTH Aachen University.

after some communication, each non-faulty process is required to decide (termination) on the same private output value (agreement), so that all processes that decide choose some process's private input value (validity). In its uniform version, however, agreement requires all processes that decide (faulty or non-faulty) to decide the same value. Only non-faulty processes are required to terminate.

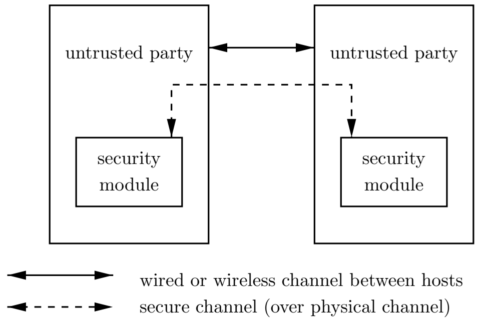

Fig. 1. Untrusted parties and security modules.

The reduction from fair exchange to consensus [2] holds in a synchronous model where each participating party is equipped with a trusted unit, that is, a tamper-proof security module like a smart card (see Fig. 1). Security modules have recently been advocated by key players in industry to improve the security of computers in the context of *trusted computing* [15]. Today, products exist which implement such trusted devices (see for example [7]). Roughly speaking, a security module is a certified piece of hardware executing a well-known algorithm. Security modules can establish confidential and authenticated channels between each other. However, since they can only communicate by exchanging messages through their (untrusted) host parties, messages may be intercepted or dropped. Overall, the security modules form a *trusted subsystem* within the overall (untrusted) system. The integrity and confidentiality of the algorithm running in the trusted subsystem is protected by the shield of tamper proof hardware. The integrity and confidentiality of data sent across the network is protected by standard cryptographic protocols. These mechanisms reduce the type and nature of adversarial behavior in the trusted subsystem to message loss and process self-destruction, two standard fault-assumptions known under the names of *omission* and *crash* in the area of fault-tolerance.

This paper proposes two novel randomized protocols for solving uniform consensus with binary inputs (and hence fair exchange) using such trusted units. Our protocols are time optimal, completing in a constant (3) expected number of rounds. They are also optimal in terms of resilience. The key insight is similar to the idea underlying the *code-division multiple access* (CDMA) communication protocol [16]: outwitting an adversary is much easier if participants share a common, secret pseudo-random number generator. In a multi-round protocol, each trusted unit can flip a coin, and take action secure in the knowledge that every other trusted unit has flipped the same value, and is taking a compatible action in that round. Because messages are encrypted, coin flip outcomes can be hidden, so dishonest parties can neither observe past coin flips nor predict future ones. (Of course, the pseudo-random algorithm itself need not be secret as long as the trusted units' common seed is kept secret, just like their common cryptographic key.) We believe that this approach is both efficient and practical.

The presentation is structured as follows. In section 2 we describe the model of computation considered, whereas in section 3 we show how to reduce fair exchange to uniform consensus. Section 4 displays related work. Optimal randomized uniform consensus protocols for binary inputs with a constant (3) expected number of rounds are introduced in sections 5 and 6. Note that both protocols may be generalized to a larger set of k values with an extra factor cost of $\log(k)$. However, we concentrate on the binary case, since we are mainly interested in solving fair exchange efficiently. Finally, we conclude with section 7, where a summary and work future directions are exhibited.

2 Model of Computation

Our model of computation is essentially synchronous: participants exchange messages in synchronous rounds. Of course, real distributed systems are not synchronous in the classical sense, but it is reasonable to assume an upper bound on how long one can expect a non-faulty processor to take before responding to a message. A processor that takes too long to join in a round is assumed to be faulty or malicious.

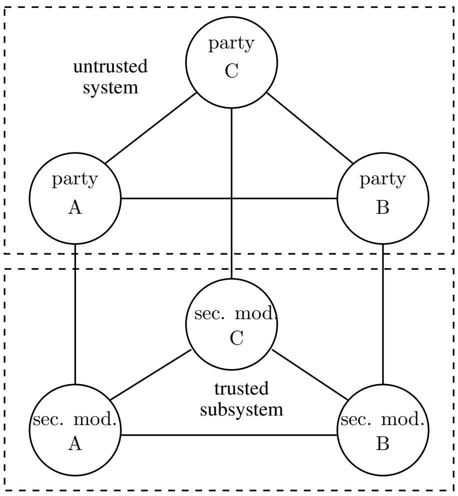

Fig. 2. The untrusted system and the trusted subsystem.

The system is logically structured into an untrusted system (including the untrusted parties and their communication channels) and the trusted subsystem consisting of the parties' individual trusted units, that is, their tamper-proof security modules (see Fig. 2). The untrusted parties can interact with their trusted units through a well-defined interface, but they cannot in any other form influence the computation within the trusted unit.

As noted, communication among the trusted units is confidential and authenticated, so malicious parties cannot interpret or tamper with these messages. Because each trusted unit sends the same encrypted message to every other trusted unit, we have receiver anonymity and so a cheating party cannot learn who is communicating to who from traffic analysis. An untrusted party can, however, prevent outgoing messages from being sent (called a *send omission*), or incoming messages from being received (called a *receive omission*) or destroy its trusted unit (called a *crash*). The effects of a crash can be regarded as a permanent send (and receive) omission.

Define a party as *cheating* if it causes send or receive omissions of its trusted unit. A party which does not cheat is *honest*. A fair exchange protocol must ensure that under no circumstances will goods be delivered to a cheating party but not to all honest parties. It is, however, acceptable to deliver the goods to all honest parties, but not to some cheating parties. Cheating may cause the exchange to *fail*, so that no goods are delivered to any party. In the absence of cheating, the exchange should *succeed*, causing goods to be delivered to all participants. For brevity, we refer to processes when we really mean untrusted processes equipped with trusted units. With a *process failure* we mean either a crash, a send message omission or a receive message omission.

3 Fair Exchange as Consensus

The reduction from fair exchange to uniform consensus works as follows. In the first round of the protocol, each party applies its acceptance test to the encrypted digital goods received from the others (in special cases this test can also be performed within the trusted unit). It then informs its trusted unit whether the goods passed the test. The trusted units broadcast this choice (using confidential and authenticated messages) within the trusted subsystem. Each unit that receives unanimous approvals starts the consensus protocol with input 1, and each trusted unit that either observes a disapproval or no message from a trusted unit starts the consensus protocol with input 0. At the end of the protocol, each trusted unit delivers the goods if the outcome of the protocol is 1, and refuses to do so if the outcome is 0.

It is easy to see that in the absence of failures or cheating all goods will be delivered. The uniform consensus protocol ensures that all honest parties agree on whether to deliver the goods, and its uniformity ensures that no trusted unit residing at a cheating party will deliver the goods if any trusted unit at an honest party decides not to. (Recall that it is acceptable if the honest processes deliver the goods after deciding 1, even if a cheating process fails to deliver the goods after deciding 0.)

As noted, the protocols considered in this paper are *randomized*, in the sense that they rely on the assumption that trusted units generate pseudo-random values that cannot be predicted by an adversary. These protocols always produce correct results, but their running time is a random variable, the so-called *Las Vegas* model. (It is straightforward to transform these protocols into *Monte-Carlo* protocols that run for a fixed number of rounds, and produce correct results with very high probability.)

To simplify the presentation, we first present an uniform consensus protocol that works in a failure model that permits send, but not receive omissions. This protocol is slightly simpler and more robust: it tolerates $f < n$ cheating processes, while the full send/receive omissions model protocol tolerates $f < n/2$ failures. Both resilience levels are optimal for their respective models [12]. Presenting the protocol in two stages illustrates how assumptions about the model affect the protocol's complexity and resilience.

4 Related Work

We build on work of Parvédy and Raynal [12]. They derive optimal early stopping deterministic uniform consensus algorithms for synchronous systems with send or send/receive omission failures. However, our algorithms are more efficient in most cases (if the number of failures is not constant) and at least comparable (otherwise).

Feldman and Micali [8] exhibit optimal consensus algorithms for Byzantine agreement, which in principle could also be used in omission failure models. Despite having also an optimal expected running time, our algorithms outperform theirs both on resilience and on the probability of not having termination violated.

Avoine, Gärtner, Guerraoui and Vukolic [2] show how to reduce the fair exchange problem in a system where processes are provided with security modules to the consensus problem in omission failure models. A solution to the fair exchange problem is presented by use of the algorithms of Parvédy and Raynal [12]. In the same context, Delporte, Fauconnier and Freiling [6] investigate solutions to consensus for *asynchronous* systems which are equipped with unreliable failure detectors. They exhibit a weak failure detector in the spirit of previous work by Chandra, Hadzilacos and Toueg [4] that allows to solve asynchronous consensus in omission failure environments.

Aspnes [1] presents a survey of randomized consensus algorithms for the shared memory model where processes are prone to crashes. These results are particularly interesting, since consensus cannot be solved deterministically in a pure asynchronous distributed system, as proved in [9] by Fischer, Lynch and Paterson.

Finally, Chaudhuri [5] introduces the k-set agreement problem, a generalization of the consensus problem, and proves it to be harder. Later, Borowsky and Gafni [3], Herlihy and Shavit [10], and Saks and Zaharoglou [14] would demonstrate that there is no wait-free protocol for k-set agreement (or consensus) in asynchronous message-passing or read/write memory models.

5 Optimal Protocol for Send Omissions

The *ConsensusS* algorithm in Figure 3 solves uniform consensus with binary inputs in optimal 3 expected synchronous rounds tolerating an optimal number of up to $f < n$ failures - send message omissions as well as process crashes.

As noted, all processes share a common secret seed and pseudo-random number generator. We denote the r-th such pseudo-random binary number by $\mathit{flip}(r)$. For each r, every process computes the same value for $\mathit{flip}(r)$.

In ConsensusS, each process broadcasts its binary input (line 1). In each subsequent round, the process waits to hear each process's *preference*. If they disagree (line 3), the process broadcasts a message informing the others. When receiving such a broadcast for the first time (line 6), every process relays it. Hence, if any non-faulty process receives mixed preferences or a $disagreement(r)$

```
1  send prefer(binary preference) to all;
2  for each round r {
3    if (both prefer(0) and prefer(1) received) {
4      send disagreement(r) to all;
5    }
6    on (receipt of disagreement(r) for the first time) {
7      send disagreement(r) to all;
8    }
9    if (all received preferences are prefer(v)) and (no disagreement(r) received){
10     if (flip(r) == v) {
11       send decide(v) to all and return(v);
12     } else {
13       send prefer(v) to all;
14     }
15   } else {
16     send prefer(flip(r)) to all;
17   }
18   if (any decide(v) received) {
19     return(v)
20   }
21 }
```

Fig. 3. Uniform consensus for send message omissions and process crashes.

message, then all processes receive a $disagreement(r)$ message and will change preference according to the coin flip. If they agree (line 9) and no message communicating disagreement seen by another process is received, then the process checks whether that preference agrees with the common pseudo-random binary number for that round. If so, it is safe to decide that value (line 11). If not, the process simply rebroadcasts the preference (line 13). If the preferences disagree or the process is informed so, then the process uses the common pseudo-random binary number to choose a new preference (line 16). If any process announces that it has decided, then the process decides on the same value (line 18).

Very informally, this protocol exploits in an essential way the observation that each process (but not the adversary) can predict the others' next coin flips. If a process receives v from all processes, then v was sent by at least one good process, so every other process will either receive all v preferences or both preferences. Any processes that receive either mixed preferences or $disagreement(r)$ messages will change preference according to the coin flip. If the coin flip is the same as v, then all processes will prefer v, and it is safe to decide.

Lemma 1. *If $f < n$, for every process the expected number of rounds of ConsensusS is 3, and the protocol terminates with probability 1.*

Proof. Think of an execution as a tree, where the root node represents the initial round and the children of a node represent the following round possibilities. Let $E(n)$ be the expected number of rounds from node n. If n has children $n.1$ and $n.2$, chosen by coin flip, then $E(n) = (1/2)(1 + E(n.1)) + (1/2)(1 + E(n.2))$. Each child contributes one plus its expected running time, but with probability one-half. Now let

- $E(n) = E_1(n)$ if at node n some non-faulty processes sent *prefer*(0) and some non-faulty processes sent *prefer*(1),

- $E(n) = E_2(n)$ if at node n all non-faulty processes sent *prefer*(v) and some non-faulty processes receive a disagreement message or both *prefer*(0) and *prefer*(1),
- $E(n) = E_3(n)$ if at node n all non-faulty processes sent *prefer*(v) and all non-faulty processes receive no disagreement messages and only *prefer*(v).

Note that if $E(n) = E_z(n)$ and $E(n.1) = E_w(n.1)$, it may be that $z \neq w$. However, it is always the case that if $E(n.1) = E_z(n.1)$ then $E(n.2) = E_z(n.2)$. The reason is that from one round to the other the values that the non-faulty processes send and receive may change. However, if the non-faulty processes behave in a way at one children, then they should behave the same way at the other, since both children just differ in the coin flip. Hence, executions differing themselves by the values sent and received by non-faulty processes may generate distinct execution trees.

Now let e be the root of an execution tree. Consider that

- $E(e) = E_1(e)$: If there are non-faulty processes that sent *prefer*(0) and other non-faulty processes that sent *prefer*(1) in round r, then at round $r+1$ every process receives at least one message *prefer*(0) and one message *prefer*(1), and thus, from round $r + 1$ on, all preference messages sent by every process (and all received as well) will be *prefer*(*flip*$(r + 1)$). Hence, all processes will decide on *flip*$(r+1)$ in the first round t such that *flip*$(t) = flip(r+1)$, and the probability that any process (and thus, a non-faulty one) violates termination is the same as the probability that such a round t never happens, that is, zero. Besides, the expected number of rounds to achieve a round t such that *flip*$(t) = flip(r+1)$ is 2. Thus, the expected number of rounds of ConsensusS is $3 = E_1(e) = (1/2)(1 + 2) + (1/2)(1 + 2)$.
- $E(e) = E_2(e)$: If all non-faulty processes sent *prefer*(v) in round r and part of the non-faulty processes receive a disagreement message or both messages *prefer*(0) and *prefer*(1) in round $r+1$, then all processes receive disagreement messages and from round $r + 1$ on, all preference messages sent by every process (and all received as well) will be *prefer*(*flip*$(r+1)$). Thus, all processes will decide on *flip*$(r + 1)$ in the first round t such that *flip*$(t) = flip(r + 1)$, and the probability that any process (and thus, a non-faulty one) violates termination is the same as the probability that such a round t never happens, that is, zero. Besides, the expected number of rounds to achieve a round t such that *flip*$(t) = flip(r + 1)$ is 2. Thus, the expected number of rounds of ConsensusS is $3 = E_2(e) = (1/2)(1 + 2) + (1/2)(1 + 2)$.
- $E(e) = E_3(e)$: If all non-faulty processes sent *prefer*(v) and receive no disagreement messages and only *prefer*(v) in round $r+1$, then if *flip*$(r+1) = v$, all non-faulty processes send *decide*(v) messages and then decide by returning v themselves. Moreover, on receipt of *decide*(v), all remaining processes decide by returning v. If *flip*$(r) \neq v$, then we fall again into the case that all non-faulty processes send *prefer*(v). That is, $E_3(e.2) = E_2(e.2) or E_3(e.2)$. Thus, the probability that any process (and thus, a non-faulty one) violates termination is zero and the expected number of rounds of ConsensusS is $3 = E_3(e) = (1/2)(1 + 1) + (1/2)(1 + 3)$.

In short, in all cases, if $f < n$, the probability that any process (and thus, a non-faulty one) violates termination is zero. Moreover, the expected number of rounds of ConsensusS is 3 for all processes. □

Lemma 2. *If $f < n$, each decided value is some process's input.*

Proof. Any decided value v is either an original input or the result of a shared coin flip. Consider the first *prefer*(*flip*(`r`)) statement to be executed, if any. In this case, there must have been a process received both *prefer*(0) and a *prefer*(1) messages, which means that some process had input value 0 and another had input value 1. It follows that either value is some process's input. □

Lemma 3. *If $f < n$, no two processes decide differently.*

Proof. Consider the first round r in which a process decides v. It must be the case that at round r, $\mathit{flip}(r) = v$ and all preference messages received by the process are *prefer*(v). As the messages from all non-faulty processes are received by all processes and there is at least one non-faulty process, all processes receive at least one *prefer*(v) message, and either decide on v at the same round r or send $\mathit{prefer}(v) = \mathit{prefer}(\mathit{flip}(r))$. It follows that from the next round $r + 1$ on, all messages sent from all processes (and thus, also all received ones) will be *prefer*(v). Henceforth, no process can decide a value different from v. □

Theorem 1. ConsensusS *solves uniform consensus with binary inputs in a synchronous system prone to crashes and send message omissions, with a probability zero of termination violation, and both an optimal constant (3) expected rounds and an optimal $n - 1$ resilience (that is, up to $n - 1$ processes may be faulty: $f < n$).*

Proof. Follows directly from Lemmas 1, 2 and 3. □

6 Optimal Protocol for Send and Receive Omissions

The *ConsensusSR* algorithm in Figure 4 solves uniform consensus with binary inputs in optimal 3 expected synchronous rounds tolerating an optimal number of up to $f < n/2$ failures - send message omissions and receive message omissions as well as process crashes.

In ConsensusSR, all processes start by broadcasting their inputs (line 2). Whenever one process does not receive a message from another, it decides that process must be faulty, and ignores it from that point on (line 6). Even so, all non-faulty processes send and receive messages from one another. Moreover, a live faulty process always receives messages from at least one non-faulty process, since otherwise, it would have less than $n/2+1$ messages and it would halt before reaching a decision (line 7).

On each round, every process checks if all received messages contain the same preferred value v (line 9). If so, it broadcasts a message that it wants to decide on v (line 10). When receiving this message for the first time (line 12), processes relay it. If a process receives such message from a majority of processes (line 15) or if it receives a message to decide on v (line 18), then it sends messages to all processes to decide on v and returns v. Note that if a non-faulty process

```
1  Recipients = set of all processes;
2  send prefer(binary preference) to all;
3  foreach round r {
4    Received(r) = set of processes from which messages were received in round r
5    Recipients = Recipients intersection Received(r);
6    Messages(r) = set of messages received in round r which were sent by Recipients;
7    if (|Messages(r)|<n/2+1) {
8      halt; // too many failures
9      if (all in Messages(r) are prefer(v)) {
10       send want_decide(v) to Recipients;
11     }
12     on (receipt of want_decide(r,v) for the first time) {
13       send want_decide(r,v) to Recipients;
14     }
15     if (want_decide(r,v) received from majority of processes) {
16       send decide(v) to all and return(v);
17     }
18     on (receipt of decide(v)) {
19       send decide(v) to all and return(v);
20     }
21     if (majority in Messages(r) are prefer(v)) {
22       send prefer(v);
23     } else {
24       send prefer(\flip(r));
25     }
26 }
```

Fig. 4. Uniform consensus for send and receive message omissions and process crashes.

relays the message, all non-faulty processes will relay the message as well, so all non-faulty processes will receive the message from a majority of processes. As every process needs a non-faulty process to relay the message in order to decide on v, if any process decides on v, then every non-faulty process does as well. If a decision is not reached, then the process either sends a message with v as its current preference (line 22), if it received a majority of preferences v, or sends a message containing $\mathit{flip}(r)$ (line 24), otherwise.

Lemma 4. *On any single round after initialization (sending the binary private input), only one value is preferred or chosen deterministically.*

Proof. A process prefers or decides v deterministically only if it sees a majority for v.

□

Lemma 5. *If $f < n/2$, for every process the expected number of rounds of ConsensusSR is 3, and the protocol terminates with probability 1.*

Proof. After initialization (sending the binary private input), if all live processes send $\mathit{prefer}(\mathit{flip}(r))$ or if all live processes send $\mathit{prefer}(v)$, they agree right away, by Lemma 4. If some send $\mathit{prefer}(v)$ and some send $\mathit{prefer}(\mathit{flip}(r))$, again by Lemma 4, then all live processes will agree in the first round t such that $v = flip(r)$, and the probability that any non-faulty process violates termination is the same as the probability that such a round t never happens, that is, zero. Besides, the expected number of rounds to achieve a round t such that $v = flip(r)$ is 2.

Once agreement by all live processes is achieved, non-faulty processes will receive a majority of $want_decide(r, v)$, send decide(v) and return(v), immediately in the same round. This is because they always receive messages from each other, that is, they always belong to the *Recipients* of non-faulty processes, so once a non-faulty process sends a $want_decide(r, v)$ message, all non-faulty processes will send $want_decide(r, v)$ messages to (and receive them from) all non-faulty processes and guarantee a majority of $want_decide(r, v)$.

In short, in all cases, if $f < n/2$, the probability that a non-faulty process violates termination is zero. Moreover, the expected number of rounds of ConsensusSR is 3 for all processes.

□

Lemma 6. *If $f < n/2$, all processes in ConsensusSR decide some process's input.*

Proof. A decided value v, from $decide(v)$, is just obtained from a $prefer(v)$. Now, by induction, a v from $prefer(v)$ has to be either an input or a $flip(r)$ for some r. However, take the first $prefer(flip(r))$ to occur, if any do. In this case, a process received both a $prefer(0)$ and a $prefer(1)$, which means that there should be a proposed input value equal to 0 and another equal to 1, as the particular $prefer(flip(r))$ was the first one to take place. Otherwise, either there would be a majority of $prefer(v)$ or Hence, $flip(r)$ must be a proposed input value if any $prefer(flip(r))$ occurs, and v must also be one of the proposed values.

□

Lemma 7. *If $f < n/2$, agreement is never violated in ConsensusSR: no two processes decide differently.*

Proof. Consider the first round r when a process decides by returning v. Then, it must be the case that a majority of $want_decide(r, v)$ is received by the process. However, because each process deciding has to receive a $want_decide(r, v)$ from a non-faulty process and non-faulty processes always receive messages from each other, when any process has a majority of $want_decide(r, v)$, it must be the case that all non-faulty processes have a majority of $want_decide(r, v)$, that is, all non-faulty processes decide by returning v as well.

□

Theorem 2. *ConsensusSR solves uniform consensus with binary inputs in a synchronous system prone to crashes, send message omissions and receive message omissions, with a probability zero of termination violation, and both an optimal constant (3) expected number of rounds and an optimal $n/2 - 1$ resilience (that is, up to $n/2 - 1$ processes may be faulty: $f < n/2$).*

Proof. Follows directly from Lemma 5, 6 and 7.

□

7 Conclusions

The key idea in this paper is that if secure coprocessors can share secret cryptographic keys (as they do), then they can also share secret seeds for secure pseudo-random number generators. Such shared coins enable randomized (Las Vegas) algorithms for fair exchange and uniform consensus that are optimal in terms of expected running time and resilience.

Both the ConsensusS and ConsensusSR binary consensus protocols can be extended to a larger set of k values in $3\log(k)$ rounds via bit-by-bit consensus. It is an open question whether faster protocols exist (perhaps by doing bit-by-bit consensus in parallel).

References

1. James Aspnes. Randomized protocols for asynchronous consensus. *Distributed Computing*, 16(2–3):165–175, September 2003.
2. Gildas Avoine, Felix Gärtner, Rachid Guerraoui, and Marko Vukolic. Gracefully degrading fair exchange with security modules. In *Proceedings of the Fifth European Dependable Computing Conference*, pages 55–71. Springer-Verlag, April 2005.
3. Elizabeth Borowsky and Eli Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *Proceedings of the Twenty-Fifth ACM Symposium on Theory of Computing*, pages 91–100. ACM Press, May 1993.
4. Tushar Deepak Chandra, Vassos Hadzilacos, and Sam Toueg. The weakest failure detector for solving consensus. *J.ACM*, 43(4):685–722, July 1996.
5. Soma Chaudhuri. Agreement is harder than consensus: Set consensus problems in totally asynchronous systems. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, pages 311–234. ACM Press, August 1990.
6. Carole Delporte-Gallet, Hugues Fauconnier, and Felix C. Freiling. Revisiting failure detection and consensus in omission failure environments. In *Proceedings of the International Colloquium on Theoretical Aspects of Computing (ICTAC05)*, Hanoi, Vietnam, October 2005.
7. Joan G. Dyer, Mark Lindemann, Ronald Perez, Reiner Sailer, Leendert van Doorn, Sean W. Smith, and Steve Weingart. Building the IBM 4758 secure coprocessor. *IEEE Computer*, 34(10):57–66, October 2001.
8. Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 148–161. ACM Press, May 1988.
9. Michael Fischer, Nancy Lynch, and Michael Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, April 1985.
10. Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *Journal of the ACM (JACM)*, 46(6):858–923, November 1999.
11. Henning Pagnia, Holger Vogt, and Felix C. Gärtner. Fair exchange. *The Computer Journal*, 46(1), 2003.
12. Philippe Raïpin Parvédy and Michel Raynal. Optimal early stopping uniform consensus in synchronous systems with process omission failures. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 302–310. ACM Press, June 2004.
13. Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreements in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, April 1980.
14. Michael Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM Journal on Computing*, 29(5):1449–1483, March 2000.
15. Trusted Computing Group. Trusted computing group homepage. Internet: `https://www.trustedcomputinggroup.org/`, 2003.
16. Andrew J. Viterbi. *CDMA : Principles of Spread Spectrum Communication*. Prentice Hall, 1995. ISBN 0201633744.

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

1987-01	*	Fachgruppe Informatik: Jahresbericht 1986
1987-02	*	David de Frutos Escrig, Klaus Indermark: Equivalence Relations of Non-Deterministic Ianov-Schemes
1987-03	*	Manfred Nagl: A Software Development Environment based on Graph Technology
1987-04	*	Claus Lewerentz, Manfred Nagl, Bernhard Westfechtel: On Integration Mechanisms within a Graph-Based Software Development Environment
1987-05	*	Reinhard Rinn: Über Eingabeanomalien bei verschiedenen Inferenzmodellen
1987-06	*	Werner Damm, Gert Döhmen: Specifying Distributed Computer Architectures in AADL*
1987-07	*	Gregor Engels, Claus Lewerentz, Wilhelm Schäfer: Graph Grammar Engineering: A Software Specification Method
1987-08	*	Manfred Nagl: Set Theoretic Approaches to Graph Grammars
1987-09	*	Claus Lewerentz, Andreas Schürr: Experiences with a Database System for Software Documents
1987-10	*	Herbert Klaeren, Klaus Indermark: A New Implementation Technique for Recursive Function Definitions
1987-11	*	Rita Loogen: Design of a Parallel Programmable Graph Reduction Machine with Distributed Memory
1987-12		J. Börstler, U. Möncke, R. Wilhelm: Table compression for tree automata
1988-01	*	Gabriele Esser, Johannes Rückert, Frank Wagner Gesellschaftliche Aspekte der Informatik
1988-02	*	Peter Martini, Otto Spaniol: Token-Passing in High-Speed Backbone Networks for Campus-Wide Environments
1988-03	*	Thomas Welzel: Simulation of a Multiple Token Ring Backbone
1988-04	*	Peter Martini: Performance Comparison for HSLAN Media Access Protocols
1988-05	*	Peter Martini: Performance Analysis of Multiple Token Rings
1988-06	*	Andreas Mann, Johannes Rückert, Otto Spaniol: Datenfunknetze
1988-07	*	Andreas Mann, Johannes Rückert: Packet Radio Networks for Data Exchange
1988-08	*	Andreas Mann, Johannes Rückert: Concurrent Slot Assignment Protocol for Packet Radio Networks
1988-09	*	W. Kremer, F. Reichert, J. Rückert, A. Mann: Entwurf einer Netzwerktopologie für ein Mobilfunknetz zur Unterstützung des öffentlichen Straßenverkehrs
1988-10	*	Kai Jakobs: Towards User-Friendly Networking
1988-11	*	Kai Jakobs: The Directory - Evolution of a Standard
1988-12	*	Kai Jakobs: Directory Services in Distributed Systems - A Survey
1988-13	*	Martine Schümmer: RS-511, a Protocol for the Plant Floor
1988-14	*	U. Quernheim: Satellite Communication Protocols - A Performance Comparison Considering On-Board Processing
1988-15	*	Peter Martini, Otto Spaniol, Thomas Welzel: File Transfer in High Speed Token Ring Networks: Performance Evaluation by Approximate Analysis and Simulation
1988-16	*	Fachgruppe Informatik: Jahresbericht 1987
1988-17	*	Wolfgang Thomas: Automata on Infinite Objects
1988-18	*	Michael Sonnenschein: On Petri Nets and Data Flow Graphs
1988-19	*	Heiko Vogler: Functional Distribution of the Contextual Analysis in Block-Structured Programming Languages: A Case Study of Tree Transducers

1988-20 * Thomas Welzel: Einsatz des Simulationswerkzeuges QNAP2 zur Leistungsbewertung von Kommunikationsprotokollen

1988-21 * Th. Janning, C. Lewerentz: Integrated Project Team Management in a Software Development Environment

1988-22 * Joost Engelfriet, Heiko Vogler: Modular Tree Transducers

1988-23 * Wolfgang Thomas: Automata and Quantifier Hierarchies

1988-24 * Uschi Heuter: Generalized Definite Tree Languages

1989-01 * Fachgruppe Informatik: Jahresbericht 1988

1989-02 * G. Esser, J. Rückert, F. Wagner (Hrsg.): Gesellschaftliche Aspekte der Informatik

1989-03 * Heiko Vogler: Bottom-Up Computation of Primitive Recursive Tree Functions

1989-04 * Andy Schürr: Introduction to PROGRESS, an Attribute Graph Grammar Based Specification Language

1989-05 J. Börstler: Reuse and Software Development - Problems, Solutions, and Bibliography (in German)

1989-06 * Kai Jakobs: OSI - An Appropriate Basis for Group Communication?

1989-07 * Kai Jakobs: ISO's Directory Proposal - Evolution, Current Status and Future Problems

1989-08 * Bernhard Westfechtel: Extension of a Graph Storage for Software Documents with Primitives for Undo/Redo and Revision Control

1989-09 * Peter Martini: High Speed Local Area Networks - A Tutorial

1989-10 * P. Davids, Th. Welzel: Performance Analysis of DQDB Based on Simulation

1989-11 * Manfred Nagl (Ed.): Abstracts of Talks presented at the WG '89 15th International Workshop on Graphtheoretic Concepts in Computer Science

1989-12 * Peter Martini: The DQDB Protocol - Is it Playing the Game?

1989-13 * Martine Schümmer: CNC/DNC Communication with MAP

1989-14 * Martine Schümmer: Local Area Networks for Manufactoring Environments with hard Real-Time Requirements

1989-15 * M. Schümmer, Th. Welzel, P. Martini: Integration of Field Bus and MAP Networks - Hierarchical Communication Systems in Production Environments

1989-16 * G. Vossen, K.-U. Witt: SUXESS: Towards a Sound Unification of Extensions of the Relational Data Model

1989-17 * J. Derissen, P. Hruschka, M.v.d. Beeck, Th. Janning, M. Nagl: Integrating Structured Analysis and Information Modelling

1989-18 A. Maassen: Programming with Higher Order Functions

1989-19 * Mario Rodriguez-Artalejo, Heiko Vogler: A Narrowing Machine for Syntax Directed BABEL

1989-20 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Graph-based Implementation of a Functional Logic Language

1990-01 * Fachgruppe Informatik: Jahresbericht 1989

1990-02 * Vera Jansen, Andreas Potthoff, Wolfgang Thomas, Udo Wermuth: A Short Guide to the AMORE System (Computing Automata, MOnoids and Regular Expressions)

1990-03 * Jerzy Skurczynski: On Three Hierarchies of Weak SkS Formulas

1990-04 R. Loogen: Stack-based Implementation of Narrowing

1990-05 H. Kuchen, A. Wagener: Comparison of Dynamic Load Balancing Strategies

1990-06 * Kai Jakobs, Frank Reichert: Directory Services for Mobile Communication

1990-07 * Kai Jakobs: What's Beyond the Interface - OSI Networks to Support Cooperative Work

1990-08 * Kai Jakobs: Directory Names and Schema - An Evaluation

1990-09 * Ulrich Quernheim, Dieter Kreuer: Das CCITT - Signalisierungssystem Nr. 7 auf Satellitenstrecken; Simulation der Zeichengabestrecke

1990-11 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Lazy Narrowing in a Graph Machine

1990-12 * Kai Jakobs, Josef Kaltwasser, Frank Reichert, Otto Spaniol: Der Computer fährt mit
1990-13 * Rudolf Mathar, Andreas Mann: Analyzing a Distributed Slot Assignment Protocol by Markov Chains
1990-14 A. Maassen: Compilerentwicklung in Miranda - ein Praktikum in funktionaler Programmierung (written in german)
1990-15 * Manfred Nagl, Andreas Schürr: A Specification Environment for Graph Grammars
1990-16 A. Schürr: PROGRESS: A VHL-Language Based on Graph Grammars
1990-17 * Marita Möller: Ein Ebenenmodell wissensbasierter Konsultationen - Unterstützung für Wissensakquisition und Erklärungsfähigkeit
1990-18 * Eric Kowalewski: Entwurf und Interpretation einer Sprache zur Beschreibung von Konsultationsphasen in Expertensystemen
1990-20 Y. Ortega Mallen, D. de Frutos Escrig: A Complete Proof System for Timed Observations
1990-21 * Manfred Nagl: Modelling of Software Architectures: Importance, Notions, Experiences
1990-22 H. Fassbender, H. Vogler: A Call-by-need Implementation of Syntax Directed Functional Programming
1991-01 Guenther Geiler (ed.), Fachgruppe Informatik: Jahresbericht 1990
1991-03 B. Steffen, A. Ingolfsdottir: Characteristic Formulae for Processes with Divergence
1991-04 M. Portz: A new class of cryptosystems based on interconnection networks
1991-05 H. Kuchen, G. Geiler: Distributed Applicative Arrays
1991-06 * Ludwig Staiger: Kolmogorov Complexity and Hausdorff Dimension
1991-07 * Ludwig Staiger: Syntactic Congruences for w-languages
1991-09 * Eila Kuikka: A Proposal for a Syntax-Directed Text Processing System
1991-10 K. Gladitz, H. Fassbender, H. Vogler: Compiler-based Implementation of Syntax-Directed Functional Programming
1991-11 R. Loogen, St. Winkler: Dynamic Detection of Determinism in Functional Logic Languages
1991-12 * K. Indermark, M. Rodriguez Artalejo (Eds.): Granada Workshop on the Integration of Functional and Logic Programming
1991-13 * Rolf Hager, Wolfgang Kremer: The Adaptive Priority Scheduler: A More Fair Priority Service Discipline
1991-14 * Andreas Fasbender, Wolfgang Kremer: A New Approximation Algorithm for Tandem Networks with Priority Nodes
1991-15 J. Börstler, A. Zündorf: Revisiting extensions to Modula-2 to support reusability
1991-16 J. Börstler, Th. Janning: Bridging the gap between Requirements Analysis and Design
1991-17 A. Zündorf, A. Schürr: Nondeterministic Control Structures for Graph Rewriting Systems
1991-18 * Matthias Jarke, John Mylopoulos, Joachim W. Schmidt, Yannis Vassiliou: DAIDA: An Environment for Evolving Information Systems
1991-19 M. Jeusfeld, M. Jarke: From Relational to Object-Oriented Integrity Simplification
1991-20 G. Hogen, A. Kindler, R. Loogen: Automatic Parallelization of Lazy Functional Programs
1991-21 * Prof. Dr. rer. nat. Otto Spaniol: ODP (Open Distributed Processing): Yet another Viewpoint
1991-22 H. Kuchen, F. Lücking, H. Stoltze: The Topology Description Language TDL
1991-23 S. Graf, B. Steffen: Compositional Minimization of Finite State Systems
1991-24 R. Cleaveland, J. Parrow, B. Steffen: The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems
1991-25 * Rudolf Mathar, Jürgen Mattfeldt: Optimal Transmission Ranges for Mobile Communication in Linear Multihop Packet Radio Networks

1991-26 M. Jeusfeld, M. Staudt: Query Optimization in Deductive Object Bases
1991-27 J. Knoop, B. Steffen: The Interprocedural Coincidence Theorem
1991-28 J. Knoop, B. Steffen: Unifying Strength Reduction and Semantic Code Motion
1991-30 T. Margaria: First-Order theories for the verification of complex FSMs
1991-31 B. Steffen: Generating Data Flow Analysis Algorithms from Modal Specifications
1992-01 Stefan Eherer (ed.), Fachgruppe Informatik: Jahresbericht 1991
1992-02 * Bernhard Westfechtel: Basismechanismen zur Datenverwaltung in strukturbezogenen Hypertextsystemen
1992-04 S. A. Smolka, B. Steffen: Priority as Extremal Probability
1992-05 * Matthias Jarke, Carlos Maltzahn, Thomas Rose: Sharing Processes: Team Coordination in Design Repositories
1992-06 O. Burkart, B. Steffen: Model Checking for Context-Free Processes
1992-07 * Matthias Jarke, Klaus Pohl: Information Systems Quality and Quality Information Systems
1992-08 * Rudolf Mathar, Jürgen Mattfeldt: Analyzing Routing Strategy NFP in Multihop Packet Radio Networks on a Line
1992-09 * Alfons Kemper, Guido Moerkotte: Grundlagen objektorientierter Datenbanksysteme
1992-10 Matthias Jarke, Manfred Jeusfeld, Andreas Miethsam, Michael Gocek: Towards a logic-based reconstruction of software configuration management
1992-11 Werner Hans: A Complete Indexing Scheme for WAM-based Abstract Machines
1992-12 W. Hans, R. Loogen, St. Winkler: On the Interaction of Lazy Evaluation and Backtracking
1992-13 * Matthias Jarke, Thomas Rose: Specification Management with CAD
1992-14 Th. Noll, H. Vogler: Top-down Parsing with Simultaneous Evaluation on Noncircular Attribute Grammars
1992-15 A. Schuerr, B. Westfechtel: Graphgrammatiken und Graphersetzungssysteme(written in german)
1992-16 * Graduiertenkolleg Informatik und Technik (Hrsg.): Forschungsprojekte des Graduiertenkollegs Informatik und Technik
1992-17 M. Jarke (ed.): ConceptBase V3.1 User Manual
1992-18 * Clarence A. Ellis, Matthias Jarke (Eds.): Distributed Cooperation in Integrated Information Systems - Proceedings of the Third International Workshop on Intelligent and Cooperative Information Systems
1992-19-00 H. Kuchen, R. Loogen (eds.): Proceedings of the 4th Int. Workshop on the Parallel Implementation of Functional Languages
1992-19-01 G. Hogen, R. Loogen: PASTEL - A Parallel Stack-Based Implementation of Eager Functional Programs with Lazy Data Structures (Extended Abstract)
1992-19-02 H. Kuchen, K. Gladitz: Implementing Bags on a Shared Memory MIMD-Machine
1992-19-03 C. Rathsack, S.B. Scholz: LISA - A Lazy Interpreter for a Full-Fledged Lambda-Calculus
1992-19-04 T.A. Bratvold: Determining Useful Parallelism in Higher Order Functions
1992-19-05 S. Kahrs: Polymorphic Type Checking by Interpretation of Code
1992-19-06 M. Chakravarty, M. Köhler: Equational Constraints, Residuation, and the Parallel JUMP-Machine
1992-19-07 J. Seward: Polymorphic Strictness Analysis using Frontiers (Draft Version)
1992-19-08 D. Gärtner, A. Kimms, W. Kluge: pi-Red^+ - A Compiling Graph-Reduction System for a Full Fledged Lambda-Calculus
1992-19-09 D. Howe, G. Burn: Experiments with strict STG code
1992-19-10 J. Glauert: Parallel Implementation of Functional Languages Using Small Processes

1992-19-11 M. Joy, T. Axford: A Parallel Graph Reduction Machine
1992-19-12 A. Bennett, P. Kelly: Simulation of Multicache Parallel Reduction
1992-19-13 K. Langendoen, D.J. Agterkamp: Cache Behaviour of Lazy Functional Programs (Working Paper)
1992-19-14 K. Hammond, S. Peyton Jones: Profiling scheduling strategies on the GRIP parallel reducer
1992-19-15 S. Mintchev: Using Strictness Information in the STG-machine
1992-19-16 D. Rushall: An Attribute Grammar Evaluator in Haskell
1992-19-17 J. Wild, H. Glaser, P. Hartel: Statistics on storage management in a lazy functional language implementation
1992-19-18 W.S. Martins: Parallel Implementations of Functional Languages
1992-19-19 D. Lester: Distributed Garbage Collection of Cyclic Structures (Draft version)
1992-19-20 J.C. Glas, R.F.H. Hofman, W.G. Vree: Parallelization of Branch-and-Bound Algorithms in a Functional Programming Environment
1992-19-21 S. Hwang, D. Rushall: The nu-STG machine: a parallelized Spineless Tagless Graph Reduction Machine in a distributed memory architecture (Draft version)
1992-19-22 G. Burn, D. Le Metayer: Cps-Translation and the Correctness of Optimising Compilers
1992-19-23 S.L. Peyton Jones, P. Wadler: Imperative functional programming (Brief summary)
1992-19-24 W. Damm, F. Liu, Th. Peikenkamp: Evaluation and Parallelization of Functions in Functional + Logic Languages (abstract)
1992-19-25 M. Kesseler: Communication Issues Regarding Parallel Functional Graph Rewriting
1992-19-26 Th. Peikenkamp: Charakterizing and representing neededness in functional loginc languages (abstract)
1992-19-27 H. Doerr: Monitoring with Graph-Grammars as formal operational Models
1992-19-28 J. van Groningen: Some implementation aspects of Concurrent Clean on distributed memory architectures
1992-19-29 G. Ostheimer: Load Bounding for Implicit Parallelism (abstract)
1992-20 H. Kuchen, F.J. Lopez Fraguas, J.J. Moreno Navarro, M. Rodriguez Artalejo: Implementing Disequality in a Lazy Functional Logic Language
1992-21 H. Kuchen, F.J. Lopez Fraguas: Result Directed Computing in a Functional Logic Language
1992-22 H. Kuchen, J.J. Moreno Navarro, M.V. Hermenegildo: Independent AND-Parallel Narrowing
1992-23 T. Margaria, B. Steffen: Distinguishing Formulas for Free
1992-24 K. Pohl: The Three Dimensions of Requirements Engineering
1992-25 * R. Stainov: A Dynamic Configuration Facility for Multimedia Communications
1992-26 * Michael von der Beeck: Integration of Structured Analysis and Timed Statecharts for Real-Time and Concurrency Specification
1992-27 W. Hans, St. Winkler: Aliasing and Groundness Analysis of Logic Programs through Abstract Interpretation and its Safety
1992-28 * Gerhard Steinke, Matthias Jarke: Support for Security Modeling in Information Systems Design
1992-29 B. Schinzel: Warum Frauenforschung in Naturwissenschaft und Technik
1992-30 A. Kemper, G. Moerkotte, K. Peithner: Object-Orientation Axiomatised by Dynamic Logic
1992-32 * Bernd Heinrichs, Kai Jakobs: Timer Handling in High-Performance Transport Systems
1992-33 * B. Heinrichs, K. Jakobs, K. Lenßen, W. Reinhardt, A. Spinner: Euro-Bridge: Communication Services for Multimedia Applications
1992-34 C. Gerlhof, A. Kemper, Ch. Kilger, G. Moerkotte: Partition-Based Clustering in Object Bases: From Theory to Practice
1992-35 J. Börstler: Feature-Oriented Classification and Reuse in IPSEN

1992-36 M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, Y. Vassiliou: Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis

1992-37 * K. Pohl, M. Jarke: Quality Information Systems: Repository Support for Evolving Process Models

1992-38 A. Zuendorf: Implementation of the imperative / rule based language PROGRES

1992-39 P. Koch: Intelligentes Backtracking bei der Auswertung funktional-logischer Programme

1992-40 * Rudolf Mathar, Jürgen Mattfeldt: Channel Assignment in Cellular Radio Networks

1992-41 * Gerhard Friedrich, Wolfgang Neidl: Constructive Utility in Model-Based Diagnosis Repair Systems

1992-42 * P. S. Chen, R. Hennicker, M. Jarke: On the Retrieval of Reusable Software Components

1992-43 W. Hans, St.Winkler: Abstract Interpretation of Functional Logic Languages

1992-44 N. Kiesel, A. Schuerr, B. Westfechtel: Design and Evaluation of GRAS, a Graph-Oriented Database System for Engineering Applications

1993-01 * Fachgruppe Informatik: Jahresbericht 1992

1993-02 * Patrick Shicheng Chen: On Inference Rules of Logic-Based Information Retrieval Systems

1993-03 G. Hogen, R. Loogen: A New Stack Technique for the Management of Runtime Structures in Distributed Environments

1993-05 A. Zündorf: A Heuristic for the Subgraph Isomorphism Problem in Executing PROGRES

1993-06 A. Kemper, D. Kossmann: Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis

1993-07 * Graduiertenkolleg Informatik und Technik (Hrsg.): Graduiertenkolleg Informatik und Technik

1993-08 * Matthias Berger: k-Coloring Vertices using a Neural Network with Convergence to Valid Solutions

1993-09 M. Buchheit, M. Jeusfeld, W. Nutt, M. Staudt: Subsumption between Queries to Object-Oriented Databases

1993-10 O. Burkart, B. Steffen: Pushdown Processes: Parallel Composition and Model Checking

1993-11 * R. Große-Wienker, O. Hermanns, D. Menzenbach, A. Pollacks, S. Repetzki, J. Schwartz, K. Sonnenschein, B. Westfechtel: Das SUKITS-Projekt: A-posteriori-Integration heterogener CIM-Anwendungssysteme

1993-12 * Rudolf Mathar, Jürgen Mattfeldt: On the Distribution of Cumulated Interference Power in Rayleigh Fading Channels

1993-13 O. Maler, L. Staiger: On Syntactic Congruences for omega-languages

1993-14 M. Jarke, St. Eherer, R. Gallersdoerfer, M. Jeusfeld, M. Staudt: ConceptBase - A Deductive Object Base Manager

1993-15 M. Staudt, H.W. Nissen, M.A. Jeusfeld: Query by Class, Rule and Concept

1993-16 * M. Jarke, K. Pohl, St. Jacobs et al.: Requirements Engineering: An Integrated View of Representation Process and Domain

1993-17 * M. Jarke, K. Pohl: Establishing Vision in Context: Towards a Model of Requirements Processes

1993-18 W. Hans, H. Kuchen, St. Winkler: Full Indexing for Lazy Narrowing

1993-19 W. Hans, J.J. Ruz, F. Saenz, St. Winkler: A VHDL Specification of a Shared Memory Parallel Machine for Babel

1993-20 * K. Finke, M. Jarke, P. Szczurko, R. Soltysiak: Quality Management for Expert Systems in Process Control

1993-21 M. Jarke, M.A. Jeusfeld, P. Szczurko: Three Aspects of Intelligent Cooperation in the Quality Cycle

1994-01 Margit Generet, Sven Martin (eds.), Fachgruppe Informatik: Jahresbericht 1993

1994-02 M. Lefering: Development of Incremental Integration Tools Using Formal Specifications
1994-03 * P. Constantopoulos, M. Jarke, J. Mylopoulos, Y. Vassiliou: The Software Information Base: A Server for Reuse
1994-04 * Rolf Hager, Rudolf Mathar, Jürgen Mattfeldt: Intelligent Cruise Control and Reliable Communication of Mobile Stations
1994-05 * Rolf Hager, Peter Hermesmann, Michael Portz: Feasibility of Authentication Procedures within Advanced Transport Telematics
1994-06 * Claudia Popien, Bernd Meyer, Axel Kuepper: A Formal Approach to Service Import in ODP Trader Federations
1994-07 P. Peters, P. Szczurko: Integrating Models of Quality Management Methods by an Object-Oriented Repository
1994-08 * Manfred Nagl, Bernhard Westfechtel: A Universal Component for the Administration in Distributed and Integrated Development Environments
1994-09 * Patrick Horster, Holger Petersen: Signatur- und Authentifikationsverfahren auf der Basis des diskreten Logarithmusproblems
1994-11 A. Schürr: PROGRES, A Visual Language and Environment for PROgramming with Graph REwrite Systems
1994-12 A. Schürr: Specification of Graph Translators with Triple Graph Grammars
1994-13 A. Schürr: Logic Based Programmed Structure Rewriting Systems
1994-14 L. Staiger: Codes, Simplifying Words, and Open Set Condition
1994-15 * Bernhard Westfechtel: A Graph-Based System for Managing Configurations of Engineering Design Documents
1994-16 P. Klein: Designing Software with Modula-3
1994-17 I. Litovsky, L. Staiger: Finite acceptance of infinite words
1994-18 G. Hogen, R. Loogen: Parallel Functional Implementations: Graphbased vs. Stackbased Reduction
1994-19 M. Jeusfeld, U. Johnen: An Executable Meta Model for Re-Engineering of Database Schemas
1994-20 * R. Gallersdörfer, M. Jarke, K. Klabunde: Intelligent Networks as a Data Intensive Application (INDIA)
1994-21 M. Mohnen: Proving the Correctness of the Static Link Technique Using Evolving Algebras
1994-22 H. Fernau, L. Staiger: Valuations and Unambiguity of Languages, with Applications to Fractal Geometry
1994-24 * M. Jarke, K. Pohl, R. Dömges, St. Jacobs, H. W. Nissen: Requirements Information Management: The NATURE Approach
1994-25 * M. Jarke, K. Pohl, C. Rolland, J.-R. Schmitt: Experience-Based Method Evaluation and Improvement: A Process Modeling Approach
1994-26 * St. Jacobs, St. Kethers: Improving Communication and Decision Making within Quality Function Deployment
1994-27 * M. Jarke, H. W. Nissen, K. Pohl: Tool Integration in Evolving Information Systems Environments
1994-28 O. Burkart, D. Caucal, B. Steffen: An Elementary Bisimulation Decision Procedure for Arbitrary Context-Free Processes
1995-01 * Fachgruppe Informatik: Jahresbericht 1994
1995-02 Andy Schürr, Andreas J. Winter, Albert Zündorf: Graph Grammar Engineering with PROGRES
1995-03 Ludwig Staiger: A Tight Upper Bound on Kolmogorov Complexity by Hausdorff Dimension and Uniformly Optimal Prediction
1995-04 Birgitta König-Ries, Sven Helmer, Guido Moerkotte: An experimental study on the complexity of left-deep join ordering problems for cyclic queries
1995-05 Sophie Cluet, Guido Moerkotte: Efficient Evaluation of Aggregates on Bulk Types
1995-06 Sophie Cluet, Guido Moerkotte: Nested Queries in Object Bases
1995-07 Sophie Cluet, Guido Moerkotte: Query Optimization Techniques Exploiting Class Hierarchies

1995-08 Markus Mohnen: Efficient Compile-Time Garbage Collection for Arbitrary Data Structures
1995-09 Markus Mohnen: Functional Specification of Imperative Programs: An Alternative Point of View of Functional Languages
1995-10 Rainer Gallersdörfer, Matthias Nicola: Improving Performance in Replicated Databases through Relaxed Coherency
1995-11 * M.Staudt, K.von Thadden: Subsumption Checking in Knowledge Bases
1995-12 * G.V.Zemanek, H.W.Nissen, H.Hubert, M.Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology
1995-13 * M.Staudt, M.Jarke: Incremental Maintenance of Externally Materialized Views
1995-14 * P.Peters, P.Szczurko, M.Jeusfeld: Oriented Information Management: Conceptual Models at Work
1995-15 * Matthias Jarke, Sudha Ram (Hrsg.): WITS 95 Proceedings of the 5th Annual Workshop on Information Technologies and Systems
1995-16 * W.Hans, St.Winkler, F.Saenz: Distributed Execution in Functional Logic Programming
1996-01 * Jahresbericht 1995
1996-02 Michael Hanus, Christian Prehofer: Higher-Order Narrowing with Definitional Trees
1996-03 * W.Scheufele, G.Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
1996-04 Klaus Pohl: PRO-ART: Enabling Requirements Pre-Traceability
1996-05 Klaus Pohl: Requirements Engineering: An Overview
1996-06 * M.Jarke, W.Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
1996-07 Olaf Chitil: The Sigma-Semantics: A Comprehensive Semantics for Functional Programs
1996-08 * S.Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
1996-09 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP96 - Fifth International Conference on Algebraic and Logic Programming
1996-09-0 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP 96 - Fifth International Conference on Algebraic and Logic Programming: Introduction and table of contents
1996-09-1 Ilies Alouini: An Implementation of Conditional Concurrent Rewriting on Distributed Memory Machines
1996-09-2 Olivier Danvy, Karoline Malmkjær: On the Idempotence of the CPS Transformation
1996-09-3 Víctor M. Gulías, José L. Freire: Concurrent Programming in Haskell
1996-09-4 Sébastien Limet, Pierre Réty: On Decidability of Unifiability Modulo Rewrite Systems
1996-09-5 Alexandre Tessier: Declarative Debugging in Constraint Logic Programming
1996-10 Reidar Conradi, Bernhard Westfechtel: Version Models for Software Configuration Management
1996-11 * C.Weise, D.Lenzkes: A Fast Decision Algorithm for Timed Refinement
1996-12 * R.Dömges, K.Pohl, M.Jarke, B.Lohmann, W.Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
1996-13 * K.Pohl, R.Klamma, K.Weidenhaupt, R.Dömges, P.Haumer, M.Jarke: A Framework for Process-Integrated Tools
1996-14 * R.Gallersdörfer, K.Klabunde, A.Stolz, M.Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996
1996-15 * H.Schimpe, M.Staudt: VAREX: An Environment for Validating and Refining Rule Bases
1996-16 * M.Jarke, M.Gebhardt, S.Jacobs, H.Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

1996-17 Manfred A. Jeusfeld, Tung X. Bui: Decision Support Components on the Internet
1996-18 Manfred A. Jeusfeld, Mike Papazoglou: Information Brokering: Design, Search and Transformation
1996-19 * P.Peters, M.Jarke: Simulating the impact of information flows in networked organizations
1996-20 Matthias Jarke, Peter Peters, Manfred A. Jeusfeld: Model-driven planning and design of cooperative information systems
1996-21 * G.de Michelis, E.Dubois, M.Jarke, F.Matthes, J.Mylopoulos, K.Pohl, J.Schmidt, C.Woo, E.Yu: Cooperative information systems: a manifesto
1996-22 * S.Jacobs, M.Gebhardt, S.Kethers, W.Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality
1996-23 * M.Gebhardt, S.Jacobs: Conflict Management in Design
1997-01 Michael Hanus, Frank Zartmann (eds.): Jahresbericht 1996
1997-02 Johannes Faassen: Using full parallel Boltzmann Machines for Optimization
1997-03 Andreas Winter, Andy Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems
1997-04 Markus Mohnen, Stefan Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler
1997-05 * S.Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge
1997-06 Matthias Nicola, Matthias Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries
1997-07 Petra Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen
1997-08 Dorothea Blostein, Andy Schürr: Computing with Graphs and Graph Rewriting
1997-09 Carl-Arndt Krapp, Bernhard Westfechtel: Feedback Handling in Dynamic Task Nets
1997-10 Matthias Nicola, Matthias Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases
1997-11 * R. Klamma, P. Peters, M. Jarke: Workflow Support for Failure Management in Federated Organizations
1997-13 Markus Mohnen: Optimising the Memory Management of Higher-Order Functional Programs
1997-14 Roland Baumann: Client/Server Distribution in a Structure-Oriented Database Management System
1997-15 George Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms
1998-01 * Fachgruppe Informatik: Jahresbericht 1997
1998-02 Stefan Gruner, Manfred Nagel, Andy Schürr: Fine-grained and Structure-Oriented Document Integration Tools are Needed for Development Processes
1998-03 Stefan Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr
1998-04 * O. Kubitz: Mobile Robots in Dynamic Environments
1998-05 Martin Leucker, Stephan Tobies: Truth - A Verification Platform for Distributed Systems
1998-06 * Matthias Oliver Berger: DECT in the Factory of the Future
1998-07 M. Arnold, M. Erdmann, M. Glinz, P. Haumer, R. Knoll, B. Paech, K. Pohl, J. Ryser, R. Studer, K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects
1998-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien
1998-10 * M. Nicola, M. Jarke: Performance Modeling of Distributed and Replicated Databases
1998-11 * Ansgar Schleicher, Bernhard Westfechtel, Dirk Jäger: Modeling Dynamic Software Processes in UML

1998-12 * W. Appelt, M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web
1998-13 Klaus Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation
1999-01 * Jahresbericht 1998
1999-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version
1999-03 * R. Gallersdörfer, M. Jarke, M. Nicola: The ADR Replication Manager
1999-04 María Alpuente, Michael Hanus, Salvador Lucas, Germán Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing
1999-05 * W. Thomas (Ed.): DLT 99 - Developments in Language Theory Fourth International Conference
1999-06 * Kai Jakobs, Klaus-Dieter Kleefeld: Informationssysteme für die angewandte historische Geographie
1999-07 Thomas Wilke: CTL+ is exponentially more succinct than CTL
1999-08 Oliver Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures
2000-01 * Jahresbericht 1999
2000-02 Jens Vöge, Marcin Jurdzinski A Discrete Strategy Improvement Algorithm for Solving Parity Games
2000-03 D. Jäger, A. Schleicher, B. Westfechtel: UPGRADE: A Framework for Building Graph-Based Software Engineering Tools
2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach
2000-05 Mareike Schoop: Cooperative Document Management
2000-06 Mareike Schoop, Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling
2000-07 * Markus Mohnen, Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages
2000-08 Thomas Arts, Thomas Noll: Verifying Generic Erlang Client-Server Implementations
2001-01 * Jahresbericht 2000
2001-02 Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03 Thierry Cachat: The power of one-letter rational languages
2001-04 Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05 Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07 Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem
2001-08 Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09 Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11 Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 * Jahresbericht 2001
2002-02 Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03 Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04 Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05 Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines

2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09 Markus Mohnen: Interfaces with Default Implementations in Java
2002-10 Martin Leucker: Logics for Mazurkiewicz traces
2002-11 Jürgen Giesl, Hans Zantema: Liveness in Rewriting
2003-01 * Jahresbericht 2002
2003-02 Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03 Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates
2003-07 Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 * Fachgruppe Informatik: Jahresbericht 2003
2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 * Fachgruppe Informatik: Jahresbericht 2004
2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut

2005-09	Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10	Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11	Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture
2005-13	Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments
2005-14	Felix C. Freiling, Sukumar Ghosh: Code Stabilization
2005-15	Uwe Naumann: The Complexity of Derivative Computation
2005-16	Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)
2005-17	Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)
2005-18	Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises "Features"
2005-19	Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers
2005-20	Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.
2005-21	Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

[*] These reports are only available as a printed version.
Please contact biblio@informatik.rwth-aachen.de to obtain copies.