Brief Announcement: Efficient Perfectly Secure Communication over Synchronous Networks

K. Srinathan*† V. Vinod* C. Pandu Rangan‡*

We consider the problem of perfectly secure communication: Given a κ-connected synchronous 2-way network, the sender $\mathcal{S}$ wants to send a message to the receiver $\mathcal{R}$ such that the intermediate players get *no* information about the message transmitted and $\mathcal{R}$ receives the message correctly. In [1] it was proved that perfect communication between all pairs of nodes (players) is possible iff $t < \frac{\kappa}{2}$, where up to t among the set of n players could be *actively* corrupt. Further, they provide protocols with optimum fault-tolerance that communicate $O(\kappa^2 t\ell)$ bits for an ℓ-bit secret message. This work improves the bit-complexity to $O(\max(\kappa^2 \log t, \kappa\ell))$ bits.

We first see why the protocol of [1] takes $O(t)$ rounds. Consider the $(2t+1)$ disjoint paths, $p_1, p_2, \ldots, p_{2t+1}$, from $\mathcal{S}$ to $\mathcal{R}$. To send a message $m \in \mathcal{F}$(a finite field), use a $(t, 2t+1)$-Shamir secret sharing scheme to get $(2t+1)$ shares $\rho_1, \rho_2, \ldots, \rho_{2t+1}$ for a random secret ρ and send ρ_i along p_i. $\mathcal{R}$ receives $\rho'_1, \rho'_2, \ldots, \rho'_{2t+1}$. If he is able to interpolate ρ then there is a common secret key (viz., ρ) between $\mathcal{S}$ and $\mathcal{R}$. Hence m can be xor-ed with ρ and *reliably* sent (i.e. sent along all paths) to $\mathcal{R}$. $\mathcal{R}$ re-xor's to get m. If the receiver is unable to interpolate ρ, then at least one i exists such that $\rho_i \neq \rho'_i$. In this case, the receiver *reliably* sends back all the $2t+1$ shares to $\mathcal{S}$, who can then find at least one path that is faulty, *reliably* inform the faulty path(s) to $\mathcal{R}$ and eliminate the faulty path(s) from subsequent execution. $\mathcal{S}$ again tries to send a random pad on this smaller number of paths and eventually succeeds. Since in the worst case, exactly one path may get eliminated from the subsequent protocol, the adversary can force the protocol to run for $O(t)$ rounds. Note that each round had a communication complexity of $O(t^2 \log |\mathcal{F}|)$ bits. The main motivation behind our protocol is the following: after the first faulty path elimination, $2t$ paths remain of which $(t-1)$ are faulty. This is more than sufficient connectivity. Thus, we can force the adversary to corrupt more than one path in the next run. In general, we ask the following question: given n paths of which t could be faulty, what is the maximum number c such that the adversary is forced to corrupt at least $c+1$ paths to trigger another round of the protocol? The answer is $(n-2t-1)$.

THEOREM 1. *Let $n = 2t+c+1$, where t is an upper bound on the number of faulty paths and n is the total number of paths. Then by sending on each path a share of a polynomial of degree $\lfloor \frac{n-1}{2} \rfloor$, the protocol of [1] fails to complete in a single phase only if the adversary reveals c faulty paths.*

The proof of the theorem is an easy generalization of the informal discussion of how the protocol of [1] works. The above idea can be used in the scheme of [1] because the error-correction/detection can be efficiently done.

CLAIM 1.1. *If the protocol does not end after k rounds, the number of faulty paths left is at most $(2^{k+1}-1)t-(2^k-1)\kappa$ and the total number of paths being $(2^{k+1}-2)t-(2^k-2)\kappa$.*

The proof is by induction on k. The claim is clearly true at the beginning of the protocol ($k=0$). Assume that after k rounds, there are $\kappa' = (2^{k+1}-2)t-(2^k-2)\kappa$ paths, $t' = (2^{k+1}-1)t-(2^k-1)\kappa$ of them being faulty. It is possible to correct at most $c = \kappa' - 2t' - 1 = 2^k\kappa - 2^{k+1}t - 1$ errors by Theorem 1. If the adversary corrupts at most c paths, it is possible for $\mathcal{R}$ to correct the errors and therefore the protocol will terminate. Therefore, the adversary must corrupt at least $c+1$ paths. The number of paths left will be $(2^{k+1}-2)t-(2^k-2)\kappa-(c+1) = (2^{k+2}-2)t-(2^{k+1}-2)\kappa$, with $(2^{k+1}-1)t-(2^k-1)\kappa-(c+1) = (2^{k+2}-1)t-(2^{k+1}-1)\kappa$ of them being faulty. □

All the faulty paths are eliminated when $k = \log(\frac{\kappa-t}{\kappa-2t})$. The protocol continues for one more round. The round complexity of the protocol is therefore, $O(\log(\frac{\kappa-t}{\kappa-2t}))$. In each round of the protocol, $O(\kappa^2)$ messages are transmitted. The overall message complexity of the protocol is $O(\kappa^2 \log(\frac{\kappa-t}{\kappa-2t}))$ $= O(\kappa^2 \log t)$ and the bit complexity is $O(\kappa^2 \ell \log t)$. Since the protocol detects faulty paths and removes them, we will encounter less faults when the next message is sent. Thus, to send a message of length ℓ, we send the ℓ bits of the message individually. The bit-complexity of sending ℓ bits is thus $O(\kappa^2 \log t + \kappa\ell)$.

†Financial support from Infosys Technologies Limited, India, is acknowledged.

‡This author would like to thank Prof. Kwangjo Kim, ICU, South Korea, for several helpful discussions. The financial support from the Ministry of Information Technology, Government of South Korea is also warmly acknowledged.

*Department of Computer Science and Engineering, Indian Institute of Technology, Madras, Chennai-600036, INDIA. e-mail:{ksrinath,vinodv}@cs.iitm.ernet.in, prangan@iitm.ac.in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PODC'03, July 13–16, 2003, Boston, Massachusetts, USA.
Copyright 2003 ACM 1-58113-708-7/03/0007...$5.00.

1. REFERENCES

[1] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *JACM*, 40(1):17–47, 1993.