Moreover, as all honest verifiers in steps $\{4, 5, \ldots, s^* - 1\}$ have signed $H(B_\ell^r)$, there does not exist a step $s' \leq s^*$ with $s' - 2 \equiv 0 \mod 3$ such that t_H $(r, s'-1)$-verifiers have signed some $v'' \neq H(B_\ell^r)$ —indeed, $|MSV^{r,s'-1}| < t_H$. Accordingly, no verifier in HSV^{r,s^*} stops with $B^r \neq B_\epsilon^r$ and $B^r \neq B_\ell^r$. That is, if a player $i \in HSV^{r,s^*}$ has stopped without propagating anything, he must have set $B^r = B_\ell^r$.

If a player $i \in HSV^{r,s^*}$ has waited time t_{s^*} and propagated a message at time $\beta_i^{r,s^*} = \alpha_i^{r,s^*} + t_{s^*}$, he has received all messages from HSV^{r,s^*-1}, including at least $t_H - |MSV^{r,s^*-1}|$ of them for 0 and v. If i has seen $> 2/3$ majority for 1, then he has seen more than $2(t_H - |MSV^{r,s^*-1}|)$ valid $(r, s^* - 1)$-messages for 1, with more than $2t_H - 3|MSV^{r,s^*-1}|$ of them from honest $(r, s^* - 1)$-verifiers. However, this implies $|HSV^{r,s^*-1}| \geq t_H - |MSV^{r,s^*-1}| + 2t_H - 3|MSV^{r,s^*-1}| > 2n - 4|MSV^{r,s^*-1}|$, contradicting the fact that

$$|HSV^{r,s^*-1}| + 4|MSV^{r,s^*-1}| < 2n,$$

which comes from the relationships for the parameters. Accordingly, i does not see $> 2/3$ majority for 1, and he sets $b_i = 0$ because Step s^* is a Coin-Fixed-To-0 step. As we have seen, $v_i = H(B_\ell^r)$. Thus i propagates $(ESIG_i(0), ESIG_i(H(B_\ell^r)), \sigma_i^{r,s})$ as we wanted to show.

For Step $s^* + 1$, since player i' has helped propagating the messages in his $CERT^r$ on or before time $\alpha_{i'}^{r,s^*} + t_{s^*}$, all honest verifiers in HSV^{r,s^*+1} have received at least t_H valid $(r, s^* - 1)$-messages for bit 0 and value $H(B_\ell^r)$ on or before they are done waiting. Furthermore, verifiers in HSV^{r,s^*+1} will not stop before receiving those (r, s^*-1)-messages, because there do not exist any other t_H valid $(r, s'-1)$-messages for bit 1 with $s' - 2 \equiv 1 \mod 3$ and $6 \leq s' \leq s^* + 1$, by the definition of Step s^*. In particular, Step $s^* + 1$ itself is a Coin-Fixed-To-1 step, but no honest verifier in HSV^{r,s^*} has propagated a message for 1, and $|MSV^{r,s^*}| < t_H$.

Thus all honest verifiers in HSV^{r,s^*+1} stop without propagating anything and set $B^r = B_\ell^r$: as before, they have received $m_\ell^{r,1}$ before they receive the desired (r, s^*-1)-messages.[20] The same can be said for all honest verifiers in future steps and all honest users in general. In particular, they all know $B^r = B_\ell^r$ within the time interval I^{r+1} and

$$T^{r+1} \leq \alpha_{i'}^{r,s^*} + t_{s^*} \leq T^r + \lambda + t_{s^*}.$$

Case 2.1.b. *Event E.b happens and there exists an honest verifier $i' \in HSV^{r,s^*}$ who should also stop without propagating anything.*

In this case we have $s^* - 2 \equiv 1 \mod 3$ and Step s^* is a Coin-Fixed-To-1 step. The analysis is similar to Case 2.1.a and many details have been omitted.

[20]If ℓ is malicious, he might send out $m_\ell^{r,1}$ late, hoping that some honest users/verifiers have not received $m_\ell^{r,1}$ yet when they receive the desired certificate for it. However, since verifier $\hat{i} \in HSV^{r,4}$ has set $b_{\hat{i}} = 0$ and $v_{\hat{i}} = H(B_\ell^r)$, as before we have that more than half of honest verifiers $i \in HSV^{r,3}$ have set $v_i = H(B_\ell^r)$. This further implies more than half of honest verifiers $i \in HSV^{r,2}$ have set $v_i = H(B_\ell^r)$, and those $(r, 2)$-verifiers have all received $m_\ell^{r,1}$. As the Adversary cannot distinguish a verifier from a non-verifier, he cannot target the propagation of $m_\ell^{r,1}$ to $(r, 2)$-verifiers without having the non-verifiers seeing it. In fact, with high probability, more than half (or a good constant fraction) of all honest users have seen $m_\ell^{r,1}$ after waiting for t_2 from the beginning of their own round r. From here on, the time λ' needed for $m_\ell^{r,1}$ to reach the remaining honest users is much smaller than Λ, and for simplicity we do not write it out in the analysis. If $4\lambda \geq \lambda'$ then the analysis goes through without any change: by the end of Step 4, all honest users would have received $m_\ell^{r,1}$. If the size of the block becomes enormous and $4\lambda < \lambda'$, then in Steps 3 and 4, the protocol could ask each verifier to wait for $\lambda'/2$ rather than 2λ, and the analysis continues to hold.

As before, player i' must have received at least t_H valid (r, s^*-1)-messages of the form $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s^*-1})$. Again by the definition of s^*, there does not exist a step $5 \leq s' < s^*$ with $s'-2 \equiv 0 \mod 3$, where at least t_H $(r, s'-1)$-verifiers have signed 0 and the same v. Thus player i' stops without propagating anything; sets $B^r = B_\epsilon^r$; and sets his own $CERT^r$ to be the set of valid (r, s^*-1)-messages for bit 1 that he has received. Moreover, any other verifier $i \in HSV^{r,s^*}$ has either stopped with $B^r = B_\epsilon^r$, or has set $b_i = 1$ and propagated $(ESIG_i(1), ESIG_i(v_i), \sigma_i^{r,s^*})$. Since player i' has helped propagating the (r, s^*-1)-messages in his $CERT^r$ by time $\alpha_{i'}^{r,s^*} + t_{s^*}$, again all honest verifiers in HSV^{r,s^*+1} stop without propagating anything and set $B^r = B_\epsilon^r$. Similarly, all honest users know $B^r = B_\epsilon^r$ within the time interval I^{r+1} and

$$T^{r+1} \leq \alpha_{i'}^{r,s^*} + t_{s^*} \leq T^r + \lambda + t_{s^*}.$$

Case 2.2.a. *Event E.a happens and there does not exist an honest verifier $i' \in HSV^{r,s^*}$ who should also stop without propagating anything.*

In this case, note that player i^* could have a valid $CERT_{i^*}^r$ consisting of the t_H desired (r, s^*-1)-messages the Adversary is able to collect or generate. However, the malicious verifiers may not help propagating those messages, so we cannot conclude that the honest users will receive them in time λ. In fact, $|MSV^{r,s^*-1}|$ of those messages may be from malicious (r, s^*-1)-verifiers, who did not propagate their messages at all and only send them to the malicious verifiers in step s^*.

Similar to Case 2.1.a, here we have $s^*-2 \equiv 0 \mod 3$, Step s^* is a Coin-Fixed-To-0 step, and the (r, s^*-1)-messages in $CERT_{i^*}^r$ are for bit 0 and $v = H(B_\ell^r)$. Indeed, all honest (r, s^*-1)-verifiers sign v, thus the Adversary cannot generate t_H valid (r, s^*-1)-messages for a different v'.

Moreover, all honest (r, s^*)-verifiers have waited time t_{s^*} and do not see $> 2/3$ majority for bit 1, again because $|HSV^{r,s^*-1}| + 4|MSV^{r,s^*-1}| < 2n$. Thus every honest verifier $i \in HSV^{r,s^*}$ sets $b_i = 0$, $v_i = H(B_\ell^r)$ by the majority vote, and propagates $m_i^{r,s^*} = (ESIG_i(0), ESIG_i(H(B_\ell^r)), \sigma_i^{r,s^*})$ at time $\alpha_i^{r,s^*} + t_{s^*}$.

Now consider the honest verifiers in Step s^*+1 (which is a Coin-Fixed-To-1 step). If the Adversary actually sends the messages in $CERT_{i^*}^r$ to some of them and causes them to stop, then similar to Case 2.1.a, all honest users know $B^r = B_\ell^r$ within the time interval I^{r+1} and

$$T^{r+1} \leq T^r + \lambda + t_{s^*+1}.$$

Otherwise, all honest verifiers in Step s^*+1 have received all the (r, s^*)-messages for 0 and $H(B_\ell^r)$ from HSV^{r,s^*} after waiting time t_{s^*+1}, which leads to $> 2/3$ majority, because $|HSV^{r,s^*}| > 2|MSV^{r,s^*}|$. Thus all the verifiers in HSV^{r,s^*+1} propagate their messages for 0 and $H(B_\ell^r)$ accordingly. Note that the verifiers in HSV^{r,s^*+1} do not stop with $B^r = B_\ell^r$, because Step s^*+1 is not a Coin-Fixed-To-0 step.

Now consider the honest verifiers in Step s^*+2 (which is a Coin-Genuinely-Flipped step). If the Adversary sends the messages in $CERT_{i^*}^r$ to some of them and causes them to stop, then again all honest users know $B^r = B_\ell^r$ within the time interval I^{r+1} and

$$T^{r+1} \leq T^r + \lambda + t_{s^*+2}.$$

Otherwise, all honest verifiers in Step $s^* + 2$ have received all the $(r, s^* + 1)$-messages for 0 and $H(B^r_\ell)$ from HSV^{r,s^*+1} after waiting time t_{s^*+2}, which leads to $> 2/3$ majority. Thus all of them propagate their messages for 0 and $H(B^r_\ell)$ accordingly: that is they do not "flip a coin" in this case. Again, note that they do not stop without propagating, because Step $s^* + 2$ is not a Coin-Fixed-To-0 step.

Finally, for the honest verifiers in Step $s^* + 3$ (which is another Coin-Fixed-To-0 step), all of them would have received at least t_H valid messages for 0 and $H(B^r_\ell)$ from HSV^{s^*+2}, if they really wait time t_{s^*+3}. Thus, whether or not the Adversary sends the messages in $CERT^r_{i^*}$ to any of them, all verifiers in HSV^{r,s^*+3} stop with $B^r = B^r_\ell$, without propagating anything. Depending on how the Adversary acts, some of them may have their own $CERT^r$ consisting of those $(r, s^* - 1)$-messages in $CERT^r_{i^*}$, and the others have their own $CERT^r$ consisting of those $(r, s^* + 2)$-messages. In any case, all honest users know $B^r = B^r_\ell$ within the time interval I^{r+1} and

$$T^{r+1} \leq T^r + \lambda + t_{s^*+3}.$$

Case 2.2.b. *Event E.b happens and there does not exist an honest verifier $i' \in HSV^{r,s^*}$ who should also stop without propagating anything.*

The analysis in this case is similar to those in Case 2.1.b and Case 2.2.a, thus many details have been omitted. In particular, $CERT^r_{i^*}$ consists of the t_H desired $(r, s^* - 1)$-messages for bit 1 that the Adversary is able to collect or generate, $s^* - 2 \equiv 1 \mod 3$, Step s^* is a Coin-Fixed-To-1 step, and no honest (r, s^*)-verifier could have seen $> 2/3$ majority for 0.

Thus, every verifier $i \in HSV^{r,s^*}$ sets $b_i = 1$ and propagates $m^{r,s^*}_i = (ESIG_i(1), ESIG_i(v_i), \sigma^{r,s^*}_i)$ at time $\alpha^{r,s^*}_i + t_{s^*}$. Similar to Case 2.2.a, in at most 3 more steps (i.e., the protocol reaches Step s^*+3, which is another Coin-Fixed-To-1 step), all honest users know $B^r = B^r_\epsilon$ within the time interval I^{r+1}. Moreover, T^{r+1} may be $\leq T^r + \lambda + t_{s^*+1}$, or $\leq T^r + \lambda + t_{s^*+2}$, or $\leq T^r + \lambda + t_{s^*+3}$, depending on when is the first time an honest verifier is able to stop without propagating.

Combining the four sub-cases, we have that all honest users know B^r within the time interval I^{r+1}, with

$T^{r+1} \leq T^r + \lambda + t_{s^*}$ in Cases 2.1.a and 2.1.b, and

$T^{r+1} \leq T^r + \lambda + t_{s^*+3}$ in Cases 2.2.a and 2.2.b.

It remains to upper-bound s^* and thus T^{r+1} for Case 2, and we do so by considering how many times the Coin-Genuinely-Flipped steps are actually executed in the protocol: that is, some honest verifiers actually have flipped a coin.

In particular, arbitrarily fix a Coin-Genuinely-Flipped step s' (i.e., $7 \leq s' \leq m + 2$ and $s' - 2 \equiv 2 \mod 3$), and let $\ell' \triangleq \arg\min_{j \in SV^{r,s'-1}} H(\sigma^{r,s'-1}_j)$. For now let us assume $s' < s^*$, because otherwise no honest verifier actually flips a coin in Step s', according to previous discussions.

By the definition of $SV^{r,s'-1}$, the hash value of the credential of ℓ' is also the smallest among all users in PK^{r-k}. Since the hash function is a random oracle, ideally player ℓ' is honest with probability at least h. As we will show later, even if the Adversary tries his best to predict the output of the random oracle and tilt the probability, player ℓ' is still honest with probability at least $p_h = h^2(1+h-h^2)$. Below we consider the case when that indeed happens: that is, $\ell' \in HSV^{r,s'-1}$.

Note that every honest verifier $i \in HSV^{r,s'}$ has received all messages from $HSV^{r,s'-1}$ by time $\alpha_i^{r,s'} + t_{s'}$. If player i needs to flip a coin (i.e., he has not seen $> 2/3$ majority for the same bit $b \in \{0,1\}$), then he sets $b_i = \texttt{lsb}(H(\sigma_{\ell'}^{r,s'-1}))$. If there exists another honest verifier $i' \in HSV^{r,s'}$ who has seen $> 2/3$ majority for a bit $b \in \{0,1\}$, then by Property (d) of Lemma 5.5, no honest verifier in $HSV^{r,s'}$ would have seen $> 2/3$ majority for a bit $b' \neq b$. Since $\texttt{lsb}(H(\sigma_{\ell'}^{r,s'-1})) = b$ with probability $1/2$, all honest verifiers in $HSV^{r,s'}$ reach an agreement on b with probability $1/2$. Of course, if such a verifier i' does not exist, then all honest verifiers in $HSV^{r,s'}$ agree on the bit $\texttt{lsb}(H(\sigma_{\ell'}^{r,s'-1}))$ with probability 1.

Combining the probability for $\ell' \in HSV^{r,s'-1}$, we have that the honest verifiers in $HSV^{r,s'}$ reach an agreement on a bit $b \in \{0,1\}$ with probability at least $\frac{p_h}{2} = \frac{h^2(1+h-h^2)}{2}$. Moreover, by induction on the majority vote as before, all honest verifiers in $HSV^{r,s'}$ have their v_i's set to be $H(B_\ell^r)$. Thus, once an agreement on b is reached in Step s', T^{r+1} is

$$\text{either } \leq T^r + \lambda + t_{s'+1} \text{ or } \leq T^r + \lambda + t_{s'+2},$$

depending on whether $b=0$ or $b=1$, following the analysis of Cases 2.1.a and 2.1.b. In particular, no further Coin-Genuinely-Flipped step will be executed: that is, the verifiers in such steps still check that they are the verifiers and thus wait, but they will all stop without propagating anything. Accordingly, before Step s^*, the number of times the Coin-Genuinely-Flipped steps are executed is distributed according to the random variable L^r. Letting Step s' be the last Coin-Genuinely-Flipped step according to L^r, by the construction of the protocol we have

$$s' = 4 + 3L^r.$$

When should the Adversary make Step s^* happen if he wants to delay T^{r+1} as much as possible? We can even assume that the Adversary knows the realization of L^r in advance. If $s^* > s'$ then it is useless, because the honest verifiers have already reached an agreement in Step s'. To be sure, in this case s^* would be $s'+1$ or $s'+2$, again depending on whether $b=0$ or $b=1$. However, this is actually Cases 2.1.a and 2.1.b, and the resulting T^{r+1} is exactly the same as in that case. More precisely,

$$T^{r+1} \leq T^r + \lambda + t_{s^*} \leq T^r + \lambda + t_{s'+2}.$$

If $s^* < s' - 3$ —that is, s^* is before the second-last Coin-Genuinely-Flipped step— then by the analysis of Cases 2.2.a and 2.2.b,

$$T^{r+1} \leq T^r + \lambda + t_{s^*+3} < T^r + \lambda + t_{s'}.$$

That is, the Adversary is actually making the agreement on B^r happen faster.

If $s^* = s' - 2$ or $s' - 1$ —that is, the Coin-Fixed-To-0 step or the Coin-Fixed-To-1 step immediately before Step s'— then by the analysis of the four sub-cases, the honest verifiers in Step s' do not get to flip coins anymore, because they have either stopped without propagating, or have seen $> 2/3$ majority for the same bit b. Therefore we have

$$T^{r+1} \leq T^r + \lambda + t_{s^*+3} \leq T^r + \lambda + t_{s'+2}.$$

In sum, no matter what s^* is, we have

$$\begin{aligned} & T^{r+1} \leq T^r + \lambda + t_{s'+2} = T^r + \lambda + t_{3L^r+6} \\ = \;& T^r + \lambda + (2(3L^r+6)-3)\lambda + \Lambda \\ = \;& T^r + (6L^r+10)\lambda + \Lambda, \end{aligned}$$

as we wanted to show. The worst case is when $s^* = s' - 1$ and Case 2.2.b happens.

Combining Cases 1 and 2 of the binary BA protocol, Lemma 5.3 holds. ■

5.9 Security of the Seed Q^r and Probability of An Honest Leader

It remains to prove Lemma 5.4. Recall that the verifiers in round r are taken from PK^{r-k} and are chosen according to the quantity Q^{r-1}. The reason for introducing the look-back parameter k is to make sure that, back at round $r-k$, when the Adversary is able to add new malicious users to PK^{r-k}, he cannot predict the quantity Q^{r-1} except with negligible probability. Note that the hash function is a random oracle and Q^{r-1} is one of its inputs when selecting verifiers for round r. Thus, no matter how malicious users are added to PK^{r-k}, from the Adversary's point of view each one of them is still selected to be a verifier in a step of round r with the required probability p (or p_1 for Step 1). More precisely, we have the following lemma.

Lemma 5.6. *With $k = O(\log_{1/2} F)$, for each round r, with overwhelming probability the Adversary did not query Q^{r-1} to the random oracle back at round $r-k$.*

Proof. We proceed by induction. Assume that for each round $\gamma < r$, the Adversary did not query $Q^{\gamma-1}$ to the random oracle back at round $\gamma - k$.[21] Consider the following mental game played by the Adversary at round $r-k$, trying to predict Q^{r-1}.

In Step 1 of each round $\gamma = r-k, \ldots, r-1$, given a specific $Q^{\gamma-1}$ not queried to the random oracle, by ordering the players $i \in PK^{\gamma-k}$ according to the hash values $H(SIG_i(\gamma, 1, Q^{\gamma-1}))$ increasingly, we obtain a random permutation over $PK^{\gamma-k}$. By definition, the leader ℓ^γ is the first user in the permutation and is honest with probability h. Moreover, when $PK^{\gamma-k}$ is large enough, for any integer $x \geq 1$, the probability that the first x users in the permutation are all malicious but the $(x+1)$st is honest is $(1-h)^x h$.

If ℓ^γ is honest, then $Q^\gamma = H(SIG_{\ell^\gamma}(Q^{\gamma-1}), \gamma)$. As the Adversary cannot forge the signature of ℓ^γ, Q^γ is distributed uniformly at random from the Adversary's point of view and, except with exponentially small probability,[22] was not queried to H at round $r-k$. Since each $Q^{\gamma+1}, Q^{\gamma+2}, \ldots, Q^{r-1}$ respectively is the output of H with $Q^\gamma, Q^{\gamma+1}, \ldots, Q^{r-2}$ as one of the inputs, they all look random to the Adversary and the Adversary could not have queried Q^{r-1} to H at round $r-k$.

Accordingly, the only case where the Adversary can predict Q^{r-1} with good probability at round $r-k$ is when all the leaders $\ell^{r-k}, \ldots, \ell^{r-1}$ are malicious. Again consider a round $\gamma \in \{r-k \ldots, r-1\}$ and the random permutation over $PK^{\gamma-k}$ induced by the corresponding hash values. If for some $x \geq 2$, the first $x-1$ users in the permutation are all malicious and the x-th is honest, then the Adversary has x possible choices for Q^γ: either of the form $H(SIG_i(Q^{\gamma-1}, \gamma))$, where i is one of

[21] As k is a small integer, without loss of generality one can assume that the first k rounds of the protocol are run under a safe environment and the inductive hypothesis holds for those rounds.

[22] That is, exponential in the length of the output of H. Note that this probability is way smaller than F.

the first $x-1$ malicious users, by making player i the actually leader of round γ; or $H(Q^{\gamma-1},\gamma)$, by forcing $B^\gamma = B^\gamma_\epsilon$. Otherwise, the leader of round γ will be the first honest user in the permutation and Q^{r-1} becomes unpredictable to the Adversary.

Which of the above x options of Q^γ should the Adversary pursue? To help the Adversary answer this question, in the mental game we actually make him more powerful than he actually is, as follows. First of all, in reality, the Adversary cannot compute the hash of a honest user's signature, thus cannot decide, for each Q^γ, the number $x(Q^\gamma)$ of malicious users at the beginning of the random permutation in round $\gamma+1$ induced by Q^γ. In the mental game, we give him the numbers $x(Q^\gamma)$ for free. Second of all, in reality, having the first x users in the permutation all being malicious does not necessarily mean they can all be made into the leader, because the hash values of their signatures must also be less than p_1. We have ignored this constraint in the mental game, giving the Adversary even more advantages.

It is easy to see that in the mental game, the optimal option for the Adversary, denoted by $\hat{Q}^\gamma$, is the one that produces the longest sequence of malicious users at the beginning of the random permutation in round $\gamma+1$. Indeed, given a specific Q^γ, the protocol does not depend on $Q^{\gamma-1}$ anymore and the Adversary can solely focus on the new permutation in round $\gamma+1$, which has the same distribution for the number of malicious users at the beginning. Accordingly, in each round γ, the above mentioned $\hat{Q}^\gamma$ gives him the largest number of options for $Q^{\gamma+1}$ and thus maximizes the probability that the consecutive leaders are all malicious.

Therefore, in the mental game the Adversary is following a Markov Chain from round $r-k$ to round $r-1$, with the state space being $\{0\}\cup\{x : x\geq 2\}$. State 0 represents the fact that the first user in the random permutation in the current round γ is honest, thus the Adversary fails the game for predicting Q^{r-1}; and each state $x\geq 2$ represents the fact that the first $x-1$ users in the permutation are malicious and the x-th is honest, thus the Adversary has x options for Q^γ. The transition probabilities $P(x,y)$ are as follows.

- $P(0,0)=1$ and $P(0,y)=0$ for any $y\geq 2$. That is, the Adversary fails the game once the first user in the permutation becomes honest.
- $P(x,0)=h^x$ for any $x\geq 2$. That is, with probability h^x, all the x random permutations have their first users being honest, thus the Adversary fails the game in the next round.
- For any $x\geq 2$ and $y\geq 2$, $P(x,y)$ is the probability that, among the x random permutations induced by the x options of Q^γ, the longest sequence of malicious users at the beginning of some of them is $y-1$, thus the Adversary has y options for $Q^{\gamma+1}$ in the next round. That is,

$$P(x,y) = \left(\sum_{i=0}^{y-1}(1-h)^i h\right)^x - \left(\sum_{i=0}^{y-2}(1-h)^i h\right)^x = (1-(1-h)^y)^x - (1-(1-h)^{y-1})^x.$$

Note that state 0 is the unique absorbing state in the transition matrix P, and every other state x has a positive probability of going to 0. We are interested in upper-bounding the number k of rounds needed for the Markov Chain to converge to 0 with overwhelming probability: that is, no matter which state the chain starts at, with overwhelming probability the Adversary loses the game and fails to predict Q^{r-1} at round $r-k$.

Consider the transition matrix $P^{(2)} \triangleq P\cdot P$ after two rounds. It is easy to see that $P^{(2)}(0,0)=1$ and $P^{(2)}(0,x)=0$ for any $x\geq 2$. For any $x\geq 2$ and $y\geq 2$, as $P(0,y)=0$, we have

$$P^{(2)}(x,y) = P(x,0)P(0,y) + \sum_{z\geq 2} P(x,z)P(z,y) = \sum_{z\geq 2} P(x,z)P(z,y).$$

Letting $\bar{h} \triangleq 1-h$, we have

$$P(x,y) = (1-\bar{h}^y)^x - (1-\bar{h}^{y-1})^x$$

and

$$P^{(2)}(x,y) = \sum_{z\geq 2} [(1-\bar{h}^z)^x - (1-\bar{h}^{z-1})^x][(1-\bar{h}^y)^z - (1-\bar{h}^{y-1})^z].$$

Below we compute the limit of $\frac{P^{(2)}(x,y)}{P(x,y)}$ as h goes to 1 —that is, $\bar{h}$ goes to 0. Note that the highest order of $\bar{h}$ in $P(x,y)$ is $\bar{h}^{y-1}$, with coefficient x. Accordingly,

$$\begin{aligned}
& \lim_{h\to 1} \frac{P^{(2)}(x,y)}{P(x,y)} = \lim_{\bar{h}\to 0} \frac{P^{(2)}(x,y)}{P(x,y)} = \lim_{\bar{h}\to 0} \frac{P^{(2)}(x,y)}{x\bar{h}^{y-1} + O(\bar{h}^y)} \\
= & \lim_{\bar{h}\to 0} \frac{\sum_{z\geq 2}[x\bar{h}^{z-1} + O(\bar{h}^z)][z\bar{h}^{y-1} + O(\bar{h}^y)]}{x\bar{h}^{y-1} + O(\bar{h}^y)} = \lim_{\bar{h}\to 0} \frac{2x\bar{h}^y + O(\bar{h}^{y+1})}{x\bar{h}^{y-1} + O(\bar{h}^y)} \\
= & \lim_{\bar{h}\to 0} \frac{2x\bar{h}^y}{x\bar{h}^{y-1}} = \lim_{\bar{h}\to 0} 2\bar{h} = 0.
\end{aligned}$$

When h is sufficiently close to 1,[23] we have

$$\frac{P^{(2)}(x,y)}{P(x,y)} \leq \frac{1}{2}$$

for any $x \geq 2$ and $y \geq 2$. By induction, for any $k > 2$, $P^{(k)} \triangleq P^k$ is such that

- $P^{(k)}(0,0) = 1$, $P^{(k)}(0,x) = 0$ for any $x \geq 2$, and
- for any $x \geq 2$ and $y \geq 2$,

$$\begin{aligned}
& P^{(k)}(x,y) = P^{(k-1)}(x,0)P(0,y) + \sum_{z\geq 2} P^{(k-1)}(x,z)P(z,y) = \sum_{z\geq 2} P^{(k-1)}(x,z)P(z,y) \\
\leq & \sum_{z\geq 2} \frac{P(x,z)}{2^{k-2}} \cdot P(z,y) = \frac{P^{(2)}(x,y)}{2^{k-2}} \leq \frac{P(x,y)}{2^{k-1}}.
\end{aligned}$$

As $P(x,y) \leq 1$, after $1 - \log_2 F$ rounds, the transition probability into any state $y \geq 2$ is negligible, starting with any state $x \geq 2$. Although there are many such states y, it is easy to see that

$$\lim_{y\to+\infty} \frac{P(x,y)}{P(x,y+1)} = \lim_{y\to+\infty} \frac{(1-\bar{h}^y)^x - (1-\bar{h}^{y-1})^x}{(1-\bar{h}^{y+1})^x - (1-\bar{h}^y)^x} = \lim_{y\to+\infty} \frac{\bar{h}^{y-1} - \bar{h}^y}{\bar{h}^y - \bar{h}^{y+1}} = \frac{1}{\bar{h}} = \frac{1}{1-h}.$$

Therefore each row x of the transition matrix P decreases as a geometric sequence with rate $\frac{1}{1-h} > 2$ when y is large enough, and the same holds for $P^{(k)}$. Accordingly, when k is large enough but still on the order of $\log_{1/2} F$, $\sum_{y\geq 2} P^{(k)}(x,y) < F$ for any $x \geq 2$. That is, with overwhelming probability the Adversary loses the game and fails to predict Q^{r-1} at round $r-k$. For $h \in (2/3, 1]$, a more complex analysis shows that there exists a constant C slightly larger than $1/2$, such that it suffices to take $k = O(\log_C F)$. Thus Lemma 5.6 holds. ■

Lemma 5.4. (restated) *Given Properties 1–3 for each round before r, $p_h = h^2(1+h-h^2)$ for L^r, and the leader ℓ^r is honest with probability at least p_h.*

[23]For example, $h = 80\%$ as suggested by the specific choices of parameters.

Proof. Following Lemma 5.6, the Adversary cannot predict Q^{r-1} back at round $r-k$ except with negligible probability. Note that this does not mean the probability of an honest leader is h for each round. Indeed, given Q^{r-1}, depending on how many malicious users are at the beginning of the random permutation of PK^{r-k}, the Adversary may have more than one options for Q^r and thus can increase the probability of a malicious leader in round $r+1$ —again we are giving him some unrealistic advantages as in Lemma 5.6, so as to simplify the analysis.

However, for each Q^{r-1} that was not queried to H by the Adversary back at round $r-k$, for any $x \geq 1$, with probability $(1-h)^{x-1}h$ the first honest user occurs at position x in the resulting random permutation of PK^{r-k}. When $x=1$, the probability of an honest leader in round $r+1$ is indeed h; while when $x=2$, the Adversary has two options for Q^r and the resulting probability is h^2. Only by considering these two cases, we have that the probability of an honest leader in round $r+1$ is at least $h \cdot h + (1-h)h \cdot h^2 = h^2(1+h-h^2)$ as desired.

Note that the above probability only considers the randomness in the protocol from round $r-k$ to round r. When all the randomness from round 0 to round r is taken into consideration, Q^{r-1} is even less predictable to the Adversary and the probability of an honest leader in round $r+1$ is at least $h^2(1+h-h^2)$. Replacing $r+1$ with r and shifts everything back by one round, the leader ℓ^r is honest with probability at least $h^2(1+h-h^2)$, as desired.

Similarly, in each Coin-Genuinely-Flipped step s, the "leader" of that step —that is the verifier in $SV^{r,s}$ whose credential has the smallest hash value, is honest with probability at least $h^2(1+h-h^2)$. Thus $p_h = h^2(1+h-h^2)$ for L^r and Lemma 5.4 holds. ■

6 *Algorand*$'_2$

In this section, we construct a version of *Algorand*$'$ working under the following assumption.

Honest Majority of Users Assumption: *More than 2/3 of the users in each PK^r are honest.*

In Section 8, we show how to replace the above assumption with the desired Honest Majority of Money assumption.

6.1 Additional Notations and Parameters for *Algorand*$'_2$

Notations

- $\mu \in \mathbb{Z}^+$: a pragmatic upper-bound to the number of steps that, with overwhelming probability, will actually taken in one round. (As we shall see, parameter μ controls how many ephemeral keys a user prepares in advance for each round.)
- L^r: a random variable representing the number of Bernoulli trials needed to see a 1, when each trial is 1 with probability $\frac{p_h}{2}$. L^r will be used to upper-bound the time needed to generate block B^r.
- t_H: a lower-bound for the number of honest verifiers in a step $s > 1$ of round r, such that with overwhelming probability (given n and p), there are $> t_H$ honest verifiers in $SV^{r,s}$.

Parameters

- *Relationships among various parameters.*
  - — For each step $s > 1$ of round r, n is chosen so that, with overwhelming probability,

$$|HSV^{r,s}| > t_H \quad \text{and} \quad |HSV^{r,s}| + 2|MSV^{r,s}| < 2t_H.$$

Note that the two inequalities above together imply $|HSV^{r,s}| > 2|MSV^{r,s}|$: that is, there is a 2/3 honest majority among selected verifiers.

The closer to 1 the value of h is, the smaller n needs to be. In particular, we use (variants of) Chernoff bounds to ensure the desired conditions hold with overwhelming probability.

- *Example choices of important parameters.*
  — $F = 10^{-18}$.
  — $n \approx 4000$, $t_H \approx 0.69n$, $k = 70$.

6.2 Implementing Ephemeral Keys in *Algorand*$'_2$

Recall that a verifier $i \in SV^{r,s}$ digitally signs his message $m_i^{r,s}$ of step s in round r, relative to an ephemeral public key $pk_i^{r,s}$, using an ephemeral secrete key $sk_i^{r,s}$ that he promptly destroys after using. When the number of possible steps that a round may take is capped by a given integer μ, we have already seen how to practically handle ephemeral keys. For example, as we have explained in *Algorand*$'_1$ (where $\mu = m + 3$), to handle all his possible ephemeral keys, from a round r' to a round $r' + 10^6$, i generates a pair (PMK, SMK), where PMK public master key of an identity based signature scheme, and SMK its corresponding secret master key. User i publicizes PMK and uses SMK to generate the secret key of each possible ephemeral public key (and destroys SMK after having done so). The set of i's ephemeral public keys for the relevant rounds is $S = \{i\} \times \{r', \ldots, r' + 10^6\} \times \{1, \ldots, \mu\}$. (As discussed, as the round $r' + 10^6$ approaches, i "refreshes" his pair (PMK, SMK).)

In practice, if μ is large enough, a round of *Algorand*$'_2$ will not take more than μ steps. In principle, however, there is the remote possibility that, for some round r the number of steps actually taken will exceed μ. When this happens, i would be unable to sign his message $m_i^{r,s}$ for any step $s > \mu$, because he has prepared in advance only μ secret keys for round r. Moreover, he could not prepare and publicize a new stash of ephemeral keys, as discussed before. In fact, to do so, he would need to insert a new public master key PMK' in a new block. But, should round r take more and more steps, no new blocks would be generated.

However, solutions exist. For instance, i may use the last ephemeral key of round r, $pk_i^{r,\mu}$, as follows. He generates another stash of key-pairs for round r —e.g., by (1) generating another master key pair $(\overline{PMK}, \overline{SMK})$; (2) using this pair to generate another, say, 10^6 ephemeral keys, $\overline{sk}_i^{r,\mu+1}, \ldots, \overline{sk}_i^{r,\mu+10^6}$, corresponding to steps $\mu+1, ..., \mu+10^6$ of round r; (3) using $sk_i^{r,\mu}$ to digitally sign $\overline{PMK}$ (and any (r,μ)-message if $i \in SV^{r,\mu}$), relative to $pk_i^{r,\mu}$; and (4) erasing $\overline{SMK}$ and $sk_i^{r,\mu}$. Should i become a verifier in a step $\mu + s$ with $s \in \{1, \ldots, 10^6\}$, then i digitally signs his $(r, \mu + s)$-message $m_i^{r,\mu+s}$ relative to his new key $\overline{pk}_i^{r,\mu+s} = (i, r, \mu + s)$. Of course, to verify this signature of i, others need to be certain that this public key corresponds to i's new public master key $\overline{PMK}$. Thus, in addition to this signature, i transmits his digital signature of $\overline{PMK}$ relative to $pk_i^{r,\mu}$.

Of course, this approach can be repeated, as many times as necessary, should round r continue for more and more steps! The last ephemeral secret key is used to authenticate a new master public key, and thus another stash of ephemeral keys for round r. And so on.

6.3 The Actual Protocol $Algorand'_2$

Recall again that, in each step s of a round r, a verifier $i \in SV^{r,s}$ uses his long-term public-secret key pair to produce his credential, $\sigma_i^{r,s} \triangleq SIG_i(r, s, Q^{r-1})$, as well as $SIG_i\left(Q^{r-1}\right)$ in case $s = 1$. Verifier i uses his ephemeral key pair, $(pk_i^{r,s}, sk_i^{r,s})$, to sign any other message m that may be required. For simplicity, we write $esig_i(m)$, rather than $sig_{pk_i^{r,s}}(m)$, to denote i's proper ephemeral signature of m in this step, and write $ESIG_i(m)$ instead of $SIG_{pk_i^{r,s}}(m) \triangleq (i, m, esig_i(m))$.

Step 1: Block Proposal

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step 1 of round r as soon as he has $CERT^{r-1}$, which allows i to unambiguously compute $H(B^{r-1})$ and Q^{r-1}.

- User i uses Q^{r-1} to check whether $i \in SV^{r,1}$ or not. If $i \notin SV^{r,1}$, he does nothing for Step 1.
- If $i \in SV^{r,1}$, that is, if i is a potential leader, then he does the following.

  (a) If i has seen $B^0, \ldots, B^{r-1}$ himself (any $B^j = B_\epsilon^j$ can be easily derived from its hash value in $CERT^j$ and is thus assumed "seen"), then he collects the round-r payments that have been propagated to him so far and computes a maximal payset PAY_i^r from them.

  (b) If i hasn't seen all $B^0, \ldots, B^{r-1}$ yet, then he sets $PAY_i^r = \emptyset$.

  (c) Next, i computes his "candidate block" $B_i^r = (r, PAY_i^r, SIG_i(Q^{r-1}), H(B^{r-1}))$.

  (c) Finally, i computes the message $m_i^{r,1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$, destroys his ephemeral secret key $sk_i^{r,1}$, and then propagates two messages, $m_i^{r,1}$ and $(SIG_i(Q^{r-1}), \sigma_i^{r,1})$, separately but simultaneously.[a]

[a]When i is the leader, $SIG_i(Q^{r-1})$ allows others to compute $Q^r = H(SIG_i(Q^{r-1}), r)$.

Selective Propagation

To shorten the global execution of Step 1 and the whole round, it is important that the $(r, 1)$-messages are *selectively propagated*. That is, for every user j in the system,

- For the first $(r, 1)$-message that he ever receives and successfully verifies,[a] whether it contains a block or is just a credential and a signature of Q^{r-1}, player j propagates it as usual.
- For all the other $(r, 1)$-messages that player j receives and successfully verifies, he propagates it only if the hash value of the credential it contains is the *smallest* among the hash values of the credentials contained in all $(r, 1)$-messages he has received and successfully verified so far.
- However, if j receives two different messages of the form $m_i^{r,1}$ from the same player i,[b] he discards the second one no matter what the hash value of i's credential is.

Note that, under selective propagation it is useful that each potential leader i propagates his credential $\sigma_i^{r,1}$ separately from $m_i^{r,1}$:[c] those small messages travel faster than blocks, ensure timely propagation of the $m_i^{r,1}$'s where the contained credentials have small hash values, while make those with large hash values disappear quickly.

[a]That is, all the signatures are correct and, if it is of the form $m_i^{r,1}$, both the block and its hash are valid —although j does not check whether the included payset is maximal for i or not.

[b]Which means i is malicious.

[c]We thank Georgios Vlachos for suggesting this.

Step 2: The First Step of the Graded Consensus Protocol GC

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step 2 of round r as soon as he has $CERT^{r-1}$.

- User i waits a maximum amount of time $t_2 \triangleq \lambda + \Lambda$. While waiting, i acts as follows.

  1. After waiting for time 2λ, he finds the user ℓ such that $H(\sigma_\ell^{r,1}) \leq H(\sigma_j^{r,1})$ for all credentials $\sigma_j^{r,1}$ that are part of the successfully verified $(r,1)$-messages he has received so far.[a]
  2. If he has received a block B^{r-1}, which matches the hash value $H(B^{r-1})$ contained in $CERT^{r-1}$,[b] and if he has received from ℓ a valid message $m_\ell^{r,1} = (B_\ell^r, esig_\ell(H(B_\ell^r)), \sigma_\ell^{r,1})$,[c] then i stops waiting and sets $v_i' \triangleq (H(B_\ell^r), \ell)$.
  3. Otherwise, when time t_2 runs out, i sets $v_i' \triangleq \perp$.
  4. When the value of v_i' has been set, i computes Q^{r-1} from $CERT^{r-1}$ and checks whether $i \in SV^{r,2}$ or not.
  5. If $i \in SV^{r,2}$, i computes the message $m_i^{r,2} \triangleq (ESIG_i(v_i'), \sigma_i^{r,2})$,[d] destroys his ephemeral secret key $sk_i^{r,2}$, and then propagates $m_i^{r,2}$. Otherwise, i stops without propagating anything.

[a]Essentially, user i privately decides that the leader of round r is user ℓ.

[b]Of course, if $CERT^{r-1}$ indicates that $B^{r-1} = B_\epsilon^{r-1}$, then i has already "received" B^{r-1} the moment he has $CERT^{r-1}$.

[c]Again, player ℓ's signatures and the hashes are all successfully verified, and PAY_ℓ^r in B_ℓ^r is a valid payset for round r —although i does not check whether PAY_ℓ^r is maximal for ℓ or not. If B_ℓ^r contains an empty payset, then there is actually no need for i to see B^{r-1} before verifying whether B_ℓ^r is valid or not.

[d]The message $m_i^{r,2}$ signals that player i considers the first component of v_i' to be the hash of the next block, or considers the next block to be empty.

Step 3: The Second Step of GC

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step 3 of round r as soon as he has $CERT^{r-1}$.

- User i waits a maximum amount of time $t_3 \triangleq t_2 + 2\lambda = 3\lambda + \Lambda$. While waiting, i acts as follows.
  1. If there exists a value v such that he has received at least t_H valid messages $m_j^{r,2}$ of the form $(ESIG_j(v), \sigma_j^{r,2})$, without any contradiction,[a] then he stops waiting and sets $v' = v$.
  2. Otherwise, when time t_3 runs out, he sets $v' = \perp$.
  3. When the value of v' has been set, i computes Q^{r-1} from $CERT^{r-1}$ and checks whether $i \in SV^{r,3}$ or not.
  4. If $i \in SV^{r,3}$, then i computes the message $m_i^{r,3} \triangleq (ESIG_i(v'), \sigma_i^{r,3})$, destroys his ephemeral secret key $sk_i^{r,3}$, and then propagates $m_i^{r,3}$. Otherwise, i stops without propagating anything.

[a]That is, he has not received two valid messages containing $ESIG_j(v)$ and a different $ESIG_j(\hat{v})$ respectively, from a player j. Here and from here on, except in the Ending Conditions defined later, whenever an honest player wants messages of a given form, messages contradicting each other are never counted or considered valid.

Step 4: Output of GC and The First Step of $BBA^\star$

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step 4 of round r as soon as he finishes his own Step 3.

- User i waits a maximum amount of time 2λ.[a] While waiting, i acts as follows.

  1. He computes v_i and g_i, the output of GC, as follows.
     (a) If there exists a value $v' \neq \perp$ such that he has received at least t_H valid messages $m_j^{r,3} = (ESIG_j(v'), \sigma_j^{r,3})$, then he stops waiting and sets $v_i \triangleq v'$ and $g_i \triangleq 2$.
     (b) If he has received at least t_H valid messages $m_j^{r,3} = (ESIG_j(\perp), \sigma_j^{r,3})$, then he stops waiting and sets $v_i \triangleq \perp$ and $g_i \triangleq 0$.[b]
     (c) Otherwise, when time 2λ runs out, if there exists a value $v' \neq \perp$ such that he has received at least $\lceil \frac{t_H}{2} \rceil$ valid messages $m_j^{r,j} = (ESIG_j(v'), \sigma_j^{r,3})$, then he sets $v_i \triangleq v'$ and $g_i \triangleq 1$.[c]
     (d) Else, when time 2λ runs out, he sets $v_i \triangleq \perp$ and $g_i \triangleq 0$.
  2. When the values v_i and g_i have been set, i computes b_i, the input of $BBA^\star$, as follows: $b_i \triangleq 0$ if $g_i = 2$, and $b_i \triangleq 1$ otherwise.
  3. i computes Q^{r-1} from $CERT^{r-1}$ and checks whether $i \in SV^{r,4}$ or not.
  4. If $i \in SV^{r,4}$, he computes the message $m_i^{r,4} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,4})$, destroys his ephemeral secret key $sk_i^{r,4}$, and propagates $m_i^{r,4}$. Otherwise, i stops without propagating anything.

[a]Thus, the maximum *total* amount of time since i starts his Step 1 of round r could be $t_4 \triangleq t_3 + 2\lambda = 5\lambda + \Lambda$.

[b]Whether Step (b) is in the protocol or not does not affect its correctness. However, the presence of Step (b) allows Step 4 to end in less than 2λ time if sufficiently many Step-3 verifiers have "signed $\perp$."

[c]It can be proved that the v' in this case, if exists, must be unique.

Step s, $5 \leq s \leq m+2$, $s-2 \equiv 0 \mod 3$: A Coin-Fixed-To-0 Step of $BBA^\star$

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step s of round r as soon as he finishes his own Step $s-1$.

- User i waits a maximum amount of time 2λ.[a] While waiting, i acts as follows.
  - *Ending Condition 0:* If at any point there exists a string $v \neq \perp$ and a step s' such that
    (a) $5 \leq s' \leq s$, $s'-2 \equiv 0 \mod 3$ —that is, Step s' is a Coin-Fixed-To-0 step,
    (b) i has received at least t_H valid messages $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$,[b] and
    (c) i has received a valid message $(SIG_j(Q^{r-1}), \sigma_j^{r,1})$ with j being the second component of v,

    then, i stops waiting and ends his own execution of Step s (and in fact of round r) right away without propagating anything as a (r,s)-verifier; sets $H(B^r)$ to be the first component of v; and sets his own $CERT^r$ to be the set of messages $m_j^{r,s'-1}$ of step (b) together with $(SIG_j(Q^{r-1}), \sigma_j^{r,1})$.[c]
  - *Ending Condition 1:* If at any point there exists a step s' such that
    (a') $6 \leq s' \leq s$, $s'-2 \equiv 1 \mod 3$ —that is, Step s' is a Coin-Fixed-To-1 step, and
    (b') i has received at least t_H valid messages $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s'-1})$,[d]

    then, i stops waiting and ends his own execution of Step s (and in fact of round r) right away without propagating anything as a (r,s)-verifier; sets $B^r = B_\epsilon^r$; and sets his own $CERT^r$ to be the set of messages $m_j^{r,s'-1}$ of sub-step (b').
  - If at any point he has received at least t_H valid $m_j^{r,s-1}$'s of the form $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he stops waiting and sets $b_i \triangleq 1$.
  - If at any point he has received at least t_H valid $m_j^{r,s-1}$'s of the form $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$, but they do not agree on the same v, then he stops waiting and sets $b_i \triangleq 0$.
  - Otherwise, when time 2λ runs out, i sets $b_i \triangleq 0$.
  - When the value b_i has been set, i computes Q^{r-1} from $CERT^{r-1}$ and checks whether $i \in SV^{r,s}$.
  - If $i \in SV^{r,s}$, i computes the message $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$ with v_i being the value he has computed in Step 4, destroys his ephemeral secret key $sk_i^{r,s}$, and then propagates $m_i^{r,s}$. Otherwise, i stops without propagating anything.

[a]Thus, the maximum *total* amount of time since i starts his Step 1 of round r could be $t_s \triangleq t_{s-1} + 2\lambda = (2s-3)\lambda + \Lambda$.

[b]Such a message from player j is counted even if player i has also received a message from j signing for 1. Similar things for Ending Condition 1. As shown in the analysis, this is to ensure that all honest users know $CERT^r$ within time λ from each other.

[c]User i now knows $H(B^r)$ and his own round r finishes. He just needs to wait until the actually block B^r is propagated to him, which may take some additional time. He still helps propagating messages as a generic user, but does not initiate any propagation as a (r,s)-verifier. In particular, he has helped propagating all messages in his $CERT^r$, which is enough for our protocol. Note that he should also set $b_i \triangleq 0$ for the binary BA protocol, but b_i is not needed in this case anyway. Similar things for all future instructions.

[d]In this case, it does not matter what the v_j's are.

Step s, $6 \leq s \leq m+2$, $s-2 \equiv 1 \mod 3$: A Coin-Fixed-To-1 Step of $BBA^\star$

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step s of round r as soon as he finishes his own Step $s-1$.

- User i waits a maximum amount of time 2λ. While waiting, i acts as follows.
  - *Ending Condition 0:* The same instructions as in a Coin-Fixed-To-0 step.
  - *Ending Condition 1:* The same instructions as in a Coin-Fixed-To-0 step.
  - If at any point he has received at least t_H valid $m_j^{r,s-1}$'s of the form $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he stops waiting and sets $b_i \triangleq 0$.[a]
  - Otherwise, when time 2λ runs out, i sets $b_i \triangleq 1$.
  - When the value b_i has been set, i computes Q^{r-1} from $CERT^{r-1}$ and checks whether $i \in SV^{r,s}$.
  - If $i \in SV^{r,s}$, i computes the message $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$ with v_i being the value he has computed in Step 4, destroys his ephemeral secret key $sk_i^{r,s}$, and then propagates $m_i^{r,s}$. Otherwise, i stops without propagating anything.

[a]Note that receiving t_H valid $(r, s-1)$-messages signing for 1 would mean Ending Condition 1.

Step s, $7 \leq s \leq m+2$, $s-2 \equiv 2 \mod 3$: A Coin-Genuinely-Flipped Step of $BBA^\star$

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step s of round r as soon as he finishes his own step $s-1$.

- User i waits a maximum amount of time 2λ. While waiting, i acts as follows.
  - *Ending Condition 0:* The same instructions as in a Coin-Fixed-To-0 step.
  - *Ending Condition 1:* The same instructions as in a Coin-Fixed-To-0 step.
  - If at any point he has received at least t_H valid $m_j^{r,s-1}$'s of the form $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he stops waiting and sets $b_i \triangleq 0$.
  - If at any point he has received at least t_H valid $m_j^{r,s-1}$'s of the form $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he stops waiting and sets $b_i \triangleq 1$.
  - Otherwise, when time 2λ runs out, letting $SV_i^{r,s-1}$ be the set of $(r, s-1)$-verifiers from whom he has received a valid message $m_j^{r,s-1}$, i sets $b_i \triangleq \texttt{lsb}(\min_{j \in SV_i^{r,s-1}} H(\sigma_j^{r,s-1}))$.
  - When the value b_i has been set, i computes Q^{r-1} from $CERT^{r-1}$ and checks whether $i \in SV^{r,s}$.
  - If $i \in SV^{r,s}$, i computes the message $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$ with v_i being the value he has computed in Step 4, destroys his ephemeral secret key $sk_i^{r,s}$, and then propagates $m_i^{r,s}$. Otherwise, i stops without propagating anything.

Remark. In principle, as considered in subsection 6.2, the protocol may take arbitrarily many steps in some round. Should this happens, as discussed, a user $i \in SV^{r,s}$ with $s > \mu$ has exhausted his stash of pre-generated ephemeral keys and has to authenticate his (r, s)-message $m_i^{r,s}$ by a "cascade" of ephemeral keys. Thus i's message becomes a bit longer and transmitting these longer messages will take a bit more time. Accordingly, after so many steps of a given round, the value of the parameter λ will automatically increase slightly. (But it reverts to the original λ once a new block is produced and a new round starts.)

Reconstruction of the Round-r Block by Non-Verifiers

Instructions for every user i in the system: User i starts his own round r as soon as he has $CERT^{r-1}$.

- i follows the instructions of each step of the protocol, participates the propagation of all messages, but does not initiate any propagation in a step if he is not a verifier in it.
- i ends his own round r by entering either Ending Condition 0 or Ending Condition 1 in some step, with the corresponding $CERT^r$.
- From there on, he starts his round $r+1$ while waiting to receive the actual block B^r (unless he has already received it), whose hash $H(B^r)$ has been pinned down by $CERT^r$. Again, if $CERT^r$ indicates that $B^r = B_\epsilon^r$, the i knows B^r the moment he has $CERT^r$.

6.4 Analysis of $Algorand'_2$

The analysis of $Algorand'_2$ is easily derived from that of $Algorand'_1$. Essentially, in $Algorand'_2$, with overwhelming probability, (a) all honest users agree on the same block B^r; the leader of a new block is honest with probability at least $p_h = h^2(1 + h - h^2)$.

7 Handling Offline Honest users

As we said, a honest user follows all his prescribed instructions, which include that of being online and running the protocol. This is not a major burden in Algorand, since the computation and bandwidth required from a honest user are quite modest. Yet, let us point out that Algorand can be easily modified so as to work in two models, in which honest users are allowed to be offline in great numbers.

Before discussing these two models, let us point out that, if the percentage of honest players were 95%, Algorand could still be run setting all parameters assuming instead that $h = 80\%$. Accordingly, Algorand would continue to work properly even if at most half of the honest players chose to go offline (indeed, a major case of"absenteeism"). In fact, at any point in time, at least 80% of the players online would be honest.

From Continual Participation to Lazy Honesty As we saw, $Algorand'_1$ and $Algorand'_2$ choose the look-back parameter k. Let us now show that choosing k properly large enables one to remove the Continual Participation requirement. This requirement ensures a crucial property: namely, that the underlying BA protocol $BBA^\star$ has a proper honest majority. Let us now explain how lazy honesty provides an alternative and attractive way to satisfy this property.

Recall that a user i is lazy-but-honest if (1) he follows all his prescribed instructions, when he is asked to participate to the protocol, and (2) he is asked to participate to the protocol only very rarely —e.g., once a week— with suitable advance notice, and potentially receiving significant rewards when he participates.

To allow Algorand to work with such players, it just suffices to "choose the verifiers of the current round among the users already in the system in a much earlier round." Indeed, recall that the verifiers for a round r are chosen from users in round $r-k$, and the selections are made based on the quantity Q^{r-1}. Note that a week consists of roughly 10,000 minutes, and assume that a round takes roughly (e.g., on average) 5 minutes, so a week has roughly 2,000 rounds. Assume that, at some point of time, a user i wishes to plan his time and know whether he is going to be a verifier in the coming week. The protocol now chooses the verifiers for a round r from users in round $r-k-2,000$, and the selections are based on $Q^{r-2,001}$. At round r, player i already knows the values $Q^{r-2,000}, \ldots, Q^{r-1}$, since they are actually part of the blockchain. Then, for each M between 1 and 2,000, i is a verifier in a step s of round $r+M$ if and only if

$$.H\left(SIG_i\left(r+M, s, Q^{r+M-2,001}\right)\right) \leq p \ .$$

Thus, to check whether he is going to be called to act as a verifier in the next 2,000 rounds, i must compute $\sigma_i^{M,s} = SIG_i\left(r+M, s, Q^{r+M-2,001}\right)$ for $M = 1$ to $2,000$ and for each step s, and check whether $.H(\sigma_i^{M,s}) \leq p$ for some of them. If computing a digital signature takes a millisecond, then this entire operation will take him about 1 minute of computation. If he is not selected as a verifier in any of these rounds, then he can go off-line with an "honest conscience". Had he continuously participated, he would have essentially taken 0 steps in the next 2,000 rounds anyway! If, instead, he is selected to be a verifier in one of these rounds, then he readies himself (e.g., by obtaining all the information necessary) to act as an honest verifier at the proper round.

By so acting, a lazy-but-honest potential verifier i only misses participating to the propagation of messages. But message propagation is typically robust. Moreover, the payers and the payees of recently propagated payments are expected to be online to watch what happens to their payments, and thus they will participate to message propagation, if they are honest.

8 Protocol *Algorand′* with Honest Majority of Money

We now, finally, show how to replace the Honest Majority of Users assumption with the much more meaningful Honest Majority of Money assumption. The basic idea is (in a proof-of-stake flavor) "to select a user $i \in PK^{r-k}$ to belong to $SV^{r,s}$ with a weight (i.e., decision power) proportional to the amount of money owned by i."[24]

By our HMM assumption, we can choose whether that amount should be owned at round $r-k$ or at (the start of) round r. Assuming that we do not mind continual participation, we opt for the latter choice. (To remove continual participation, we would have opted for the former choice. *Better said, for the amount of money owned at round* $r-k-2,000$.)

There are many ways to implement this idea. The simplest way would be to have each key hold at most 1 unit of money and then select at random n users i from PK^{r-k} such that $a_i^{(r)} = 1$.

[24]We should say $PK^{r-k-2,000}$ so as to replace continual participation. For simplicity, since one may wish to require continual participation anyway, we use PK^{r-k} as before, so as to carry one less parameter.

The Next Simplest Implementation

The next simplest implementation may be to demand that each public key owns a maximum amount of money M, for some fixed M. The value M is small enough compared with the total amount of money in the system, such that the probability a key belongs to the verifier set of more than one step in —say— k rounds is negligible. Then, a key $i \in PK^{r-k}$, owning an amount of money $a_i^{(r)}$ in round r, is chosen to belong to $SV^{r,s}$ if

$$.H\left(SIG_i\left(r, s, Q^{r-1}\right)\right) \leq p \cdot \frac{a_i^{(r)}}{M} \ .$$

And all proceeds as before.

A More Complex Implementation

The last implementation "forced a rich participant in the system to own many keys".

An alternative implementation, described below, generalizes the notion of status and consider each user i to consist of $K+1$ *copies* (i, v), each of which is independently selected to be a verifier, and will own his own ephemeral key $(pk_{i,v}^{r,s}, sk_{i,v}^{r,s})$ in a step s of a round r. The value K depends on the amount of money $a_i^{(r)}$ owned by i in round r.

Let us now see how such a system works in greater detail.

Number of Copies Let n be the targeted expected cardinality of each verifier set, and let $a_i^{(r)}$ be the amount of money owned by a user i at round r. Let A^r be the total amount of money owned by the users in PK^{r-k} at round r, that is,

$$A^r = \sum_{i \in PK^{r-k}} a_i^{(r)}.$$

If i is an user in PK^{r-k}, then i's copies are $(i, 1), \ldots, (i, K+1)$, where

$$K = \left\lfloor \frac{n \cdot a_i^{(r)}}{A^r} \right\rfloor \ .$$

Example. Let $n = 1{,}000$, $A^r = 10^9$, and $a_i^{(r)} = 3.7$ millions. Then,

$$K = \left\lfloor \frac{10^3 \cdot (3.7 \cdot 10^6)}{10^9} \right\rfloor = \lfloor 3.7 \rfloor = 3 \ .$$

Verifiers and Credentials Let i be a user in PK^{r-k} with $K+1$ copies.

For each $v = 1, \ldots, K$, copy (i, v) belongs to $SV^{r,s}$ automatically. That is, i's credential is $\sigma_{i,v}^{r,s} \triangleq SIG_i((i,v), r, s, Q^{r-1})$, but the corresponding condition becomes $.H(\sigma_{i,v}^{r,s}) \leq 1$, which is always true.

For copy $(i, K+1)$, for each Step s of round r, i checks whether

$$.H\big(SIG_i\big(\,(i, K+1), r, s, Q^{r-1}\,\big)\big) \leq a_i^{(r)} \frac{n}{A^r} - K \ .$$

If so, copy $(i, K+1)$ belongs to $SV^{r,s}$. To prove it, i propagates the credential

$$\sigma^{r,1}_{i,K+1} = SIG_i\big((i, K+1), r, s, Q^{r-1}\big).$$

Example. As in the previous example, let $n = 1K$, $a_i^{(r)} = 3.7M$, $A^r = 1B$, and i has 4 copies: $(i,1), \ldots, (i,4)$. Then, the first 3 copies belong to $SV^{r,s}$ automatically. For the 4th one, conceptually, *Algorand'* independently rolls a biased coin, whose probability of Heads is 0.7. Copy $(i,4)$ is selected if and only if the coin toss is Heads.

(Of course, this biased coin flip is implemented by hashing, signing, and comparing —as we have done all along in this paper— so as to enable i to prove his result.)

Business as Usual Having explained how verifiers are selected and how their credentials are computed at each step of a round r, the execution of a round is similar to that already explained.

9 Handling Forks

Having reduced the probability of forks to 10^{-12} or 10^{-18}, it is practically unnecessary to handle them in the remote chance that they occur. Algorand, however, can also employ various fork resolution procedures, with or without proof of work.

One possible way of instructing the users to resolve forks is as follows:

- Follow the longest chain if a user sees multiple chains.
- If there are more than one longest chains, follow the one with a non-empty block at the end. If all of them have empty blocks at the end, consider their second-last blocks.
- If there are more than one longest chains with non-empty blocks at the end, say the chains are of length r, follow the one whose leader of block r has the smallest credential. If there are ties, follow the one whose block r itself has the smallest hash value. If there are still ties, follow the one whose block r is ordered the first lexicographically.

10 Handling Network Partitions

As said, we assume the propagation times of messages among all users in the network are upper-bounded by λ and Λ. This is not a strong assumption, as today's Internet is fast and robust, and the actual values of these parameters are quite reasonable. Here, let us point out that $Algorand'_2$ continues to work even if the Internet occasionally got partitioned into two parts. The case when the Internet is partitioned into more than two parts is similar.

10.1 Physical Partitions

First of all, the partition may be caused by physical reasons. For example, a huge earthquake may end up completely breaking down the connection between Europe and America. In this case, the malicious users are also partitioned and there is no communication between the two parts. Thus there will be two Adversaries, one for part 1 and the other for part 2. Each Adversary still tries to break the protocol in its own part.

Assume the partition happens in the middle of round r. Then each user is still selected as a verifier based on PK^{r-k}, with the same probability as before. Let $HSV_i^{r,s}$ and $MSV_i^{r,s}$ respectively be the set of honest and malicious verifiers in a step s in part $i \in \{1, 2\}$. We have

$$|HSV_1^{r,s}| + |MSV_1^{r,s}| + |HSV_2^{r,s}| + |MSV_2^{r,s}| = |HSV^{r,s}| + |MSV^{r,s}|.$$

Note that $|HSV^{r,s}| + |MSV^{r,s}| < |HSV^{r,s}| + 2|MSV^{r,s}| < 2t_H$ with overwhelming probability.

If some part i has $|HSV_i^{r,s}| + |MSV_i^{r,s}| \geq t_H$ with non-negligible probability, e.g., 1%, then the probability that $|HSV_{3-i}^{r,s}| + |MSV_{3-i}^{r,s}| \geq t_H$ is very low, e.g., 10^{-16} when $F = 10^{-18}$. In this case, we may as well treat the smaller part as going offline, because there will not be enough verifiers in this part to generate t_H signatures to certify a block.

Let us consider the larger part, say part 1 without loss of generality. Although $|HSV^{r,s}| < t_H$ with negligible probability in each step s, when the network is partitioned, $|HSV_1^{r,s}|$ may be less than t_H with some non-negligible probability. In this case the Adversary may, with some other non-negligible probability, force the binary BA protocol into a fork in round r, with a non-empty block B^r and the empty block B_ϵ^r both having t_H valid signatures.[25] For example, in a Coin-Fixed-To-0 step s, all verifiers in $HSV_1^{r,s}$ signed for bit 0 and $H(B^r)$, and propagated their messages. All verifiers in $MSV_1^{r,s}$ also signed 0 and $H(B^r)$, but withheld their messages. Because $|HSV_1^{r,s}| + |MSV_1^{r,s}| \geq t_H$, the system has enough signatures to certify B^r. However, since the malicious verifiers withheld their signatures, the users enter step $s+1$, which is a Coin-Fixed-To-1 step. Because $|HSV_1^{r,s}| < t_H$ due to the partition, the verifiers in $HSV_1^{r,s+1}$ did not see t_H signatures for bit 0 and they all signed for bit 1. All verifiers in $MSV_1^{r,s+1}$ did the same. Because $|HSV_1^{r,s+1}| + |MSV_1^{r,s+1}| \geq t_H$, the system has enough signatures to certify B_ϵ^r. The Adversary then creates a fork by releasing the signatures of $MSV_1^{r,s}$ for 0 and $H(B^r)$.

Accordingly, there will be two Q^r's, defined by the corresponding blocks of round r. However, the fork will not continue and only one of the two branches may grow in round $r+1$.

Additional Instructions for *Algorand$'_2$*. *When seeing a non-empty block B^r and the empty block B_ϵ^r, follow the non-empty one (and the Q^r defined by it).*

Indeed, by instructing the users to go with the non-empty block in the protocol, if a large amount of honest users in PK^{r+1-k} realize there is a fork at the beginning of round $r+1$, then the empty block will not have enough followers and will not grow. Assume the Adversary manages to partition the honest users so that some honest users see B^r (and perhaps B_ϵ^r), and some only see B_ϵ^r. Because the Adversary cannot tell which one of them will be a verifier following B^r and which will be a verifier following B_ϵ^r, the honest users are randomly partitioned and each one of them still becomes a verifier (either with respect to B^r or with respect to B_ϵ^r) in a step $s > 1$ with probability p. For the malicious users, each one of them may have two chances to become a verifier, one with B^r and the other with B_ϵ^r, each with probability p independently.

Let $HSV_{1;B^r}^{r+1,s}$ be the set of honest verifiers in step s of round $r+1$ following B^r. Other notations such as $HSV_{1;B_\epsilon^r}^{r+1,s}$, $MSV_{1;B^r}^{r+1,s}$ and $MSV_{1;B_\epsilon^r}^{r+1,s}$ are similarly defined. By Chernoff bound, it is easy

[25]Having a fork with two non-empty blocks is not possible with or without partitions, except with negligible probability.

to see that with overwhelming probability,

$$|HSV_{1;B^r}^{r+1,s}| + |HSV_{1;B^r_\epsilon}^{r+1,s}| + |MSV_{1;B^r}^{r+1,s}| + |MSV_{1;B^r_\epsilon}^{r+1,s}| < 2t_H.$$

Accordingly, the two branches cannot both have t_H proper signatures certifying a block for round $r+1$ in the same step s. Moreover, since the selection probabilities for two steps s and s' are the same and the selections are independent, also with overwhelming probability

$$|HSV_{1;B^r}^{r+1,s}| + |MSV_{1;B^r}^{r+1,s}| + |HSV_{1;B^r_\epsilon}^{r+1,s'}| + |MSV_{1;B^r_\epsilon}^{r+1,s'}| < 2t_H,$$

for any two steps s and s'. When $F = 10^{-18}$, by the union bound, as long as the Adversary cannot partition the honest users for a long time (say 10^4 steps, which is more than 55 hours with $\lambda = 10$ seconds[26]), with high probability (say $1-10^{-10}$) at most one branch will have t_H proper signatures to certify a block in round $r+1$.

Finally, if the physical partition has created two parts with roughly the same size, then the probability that $|HSV_i^{r,s}| + |MSV_i^{r,s}| \geq t_H$ is small for each part i. Following a similar analysis, even if the Adversary manages to create a fork with some non-negligible probability in each part for round r, at most one of the four branches may grow in round $r+1$.

10.2 Adversarial Partition

Second of all, the partition may be caused by the Adversary, so that the messages propagated by the honest users in one part will not reach the honest users in the other part directly, but the Adversary is able to forward messages between the two parts. Still, once a message from one part reaches an honest user in the other part, it will be propagated in the latter as usual. If the Adversary is willing to spend a lot of money, it is conceivable that he may be able to hack the Internet and partition it like this for a while.

The analysis is similar to that for the larger part in the physical partition above (the smaller part can be considered as having population 0): the Adversary may be able to create a fork and each honest user only sees one of the branches, but at most one branch may grow.

10.3 Network Partitions in Sum

Although network partitions can happen and a fork in one round may occur under partitions, there is no lingering ambiguity: a fork is very short-lived, and in fact lasts for at most a single round. In all parts of the partition except for at most one, the users cannot generate a new block and thus (a) realize there is a partition in the network and (b) never rely on blocks that will "vanish".

Acknowledgements

We would like to first acknowledge Sergey Gorbunov, coauthor of the cited Democoin system.

Most sincere thanks go to Maurice Herlihy, for many enlightening discussions, for pointing out that pipelining will improve Algorand's throughput performance, and for greatly improving the

[26]Note that a user finishes a step s without waiting for 2λ time only if he has seen at least t_H signatures for the same message. When there are not enough signatures, each step will last for 2λ time.

exposition of an earlier version of this paper. Many thanks to Sergio Rajsbaum, for his comments on an earlier version of this paper. Many thanks to Vinod Vaikuntanathan, for several deep discussions and insights. Many thanks to Yossi Gilad, Rotem Hamo, Georgios Vlachos, and Nickolai Zeldovich for starting to test these ideas, and for many helpful comments and discussions.

Silvio Micali would like to personally thank Ron Rivest for innumerable discussions and guidance in cryptographic research over more than 3 decades, for coauthoring the cited micropayment system that has inspired one of the verifier selection mechanisms of Algorand.

We hope to bring this technology to the next level. Meanwhile the travel and companionship are great fun, for which we are very grateful.

References

[1] *Bitcoin Computation Waste*, http://gizmodo.com/the-worlds-most-powerful-computer-network-is-being-was-50 2013.

[2] Bitcoinwiki. *Proof of Stake.* `http://www.blockchaintechnologies.com/blockchain-applications` As of 5 June 2016.

[3] Coindesk.com. *Bitcoin: A Peer-to-Peer Electronic Cash System* `http://www.coindesk.com/ibm-reveals-proof-concept-blockchain-powered-internet-things/` As of June 2016.

[4] Ethereum. *Ethereum.* `https://github.com/ethereum/`. As of 12 June 2016.

[5] HowStuffWorks.com. *How much actual money is there in the world?*, `https://money.howstuffworks.com/how-much-money-is-in-the-world.htm`. As of 5 June 2016.

[6] en.wikipedia.org/wiki/Sortition.

[7] M. Ben-Or. *Another advantage of free choice: Completely asynchronous agreement protocols.* Proc. 2nd Annual Symposium on Principles of Distributed Computing, ACM, New York, 1983, pp. 27-30.

[8] M. Castro and B. Liskov. *Practical Byzantine Fault Tolerance*, Proceedings of the Third Symposium on Operating Systems Design and Implementation. New Orleans, Louisiana, USA, 1999, pp. 173–186.

[9] D. L. Chaum, *Random Sample Elections*, `https://www.scribd.com/mobile/document/236881043/Random-Sa`

[10] B. Chor and C. Dwork. *Randomization in Byzantine agreement, in Randomness and Computation.* S. Micali, ed., JAI Press, Greenwich, CT, 1989, pp. 433-498.

[11] C. Decker and R. Wattenhofer. *Information Propagation in the Bitcoin Network.* 13-th IEEE Conference on Peer-to-Peer Computing, 2013.

[12] D. Dolev. *The Byzantine Generals Strike Again.* J. Algorithms, 3, (1982), pp. 14-30.

[13] D. Dolev and H.R. Strong. *Authenticated algorithms for Byzantine agreement.* SIAM Journal on Computing 12 (4), 656-666.

[14] C. Dwork and M. Naor. *Pricing via Processing, Or, Combatting Junk Mail.* Advances in Cryptology, CRYPTO'92: Lecture Notes in Computer Science No. 740. Springer: 139–147.

[15] P. Feldman and S. Micali. *An Optimal Probabilistic Algorithm for Synchronous Byzantine Agreement.* (Preliminary version in STOC 88.) SIAM J. on Computing, 1997.

[16] M. Fischer. *The consensus problem in unreliable distributed systems (a brief survey).* Proc. International Conference on Foundations of Computation, 1983.

[17] S. Goldwasser, S. Micali, and R. Rivest. *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attack.* SIAM Journal of Computing, 17, No. 2, April 1988, pp. 281-308

[18] S. Gorbunov and S. Micali. *Democoin: A Publicly Verifiable and Jointly Serviced Cryptocurrency.* `https://eprint.iacr.org/2015/521`, May 30, 2015.

[19] J. Katz and C-Y Koo. *On Expected Constant-Round Protocols for Byzantine Agreement.* `https://www.cs.umd.edu/~jkatz/papers/BA.pdf`.

[20] A. Kiayias, A. Russel, B. David, and R. Oliynycov.. *Ouroburos: A provably secure proof-of-stake protocol.* Cryptology ePrint Archive, Report 2016/889, 2016. http://eprint.iacr.org/2016/889.

[21] S. King and S. Nadal. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*, 2012.

[22] D. Lazar and Y. Gilad. Personal Communication.

[23] N. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, 1996.

[24] S. Micali. *Algorand: The Efficient Public Ledger.* `https://arxiv.org/abs/1607.01341`.

[25] S. Micali. *Fast And Furious Byzantine Agreement.* Innovation in Theoretical Computer Science 2017. Berkeley, CA, January 2017. Single-page abstract.

[26] S. Micali. *Byzantine Agreement, Made Trivial.* `https://people.csail.mit.edu/silvio/SelectedScientif`

[27] S. Micali, M. Rabin and S. Vadhan. *Verifiable Random Functions.* 40th Foundations of Computer Science (FOCS), New York, Oct 1999.

[28] S. Micali and R. L. Rivest. *Micropayments Revisited.* Lecture Notes in Computer Science, Vol. 2271, pp 149-163, Springer Verlag, 2002.

[29] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* `http://www.bitcoin.org/bitcoin.pdf`, May 2009.

[30] R. Pass and E. Shi. *The Sleepy Model of Consensus.* Cryptology ePrint Archive, Feb 2017, Report 2017/918.

[31] M. Pease, R. Shostak, and L. Lamport. *Reaching agreement in the presence of faults.* J. Assoc. Comput. Mach., 27 (1980), pp. 228-234.

[32] M. Rabin. *Randomized Byzantine generals.* 24th Foundations of Computer Science (FOCS), IEEE Computer Society Press, Los Alamitos, CA, 1983, pp. 403-409.

[33] R. Turpin and B. Coan. *Extending binary Byzantine agreement to multivalued Byzantine agreement.* Inform. Process. Lett., 18 (1984), pp. 73-76.