Simulating authenticated broadcast

For round-based algorithms, a protocol that provides the properties of broadcasting authenticated messages.

Authenticated broadcast

Synchronous-round based

- $p$ broadcasts $m$ in round $k$: $broadcast(p, m, k)$
- process $accept(p, m, k)$ when:
  - receives $(p, m, k)$
  - verifies $p$’s signature.

Correctness: If correct $p$ broadcasts $(p, m, k)$ in round $k$ then every correct process accepts $(p, m, k)$ in the same round.

Unforgeability: If correct $p$ does not broadcast $(p, m, k)$ then no correct process ever accepts $(p, m, k)$.

Relay: If correct $p$ accepts $(p, m, k)$ in round $r \geq k$, then every correct process accepts $(p, m, k)$ by round $r + 1$. 
Authenticated Arbitrary Consensus: I

// m ∈ {0, 1}, non-loquacious protocol

process p:
    if (p == p₁) value = m;
    else value = 0;
    for r = 1 to t + 1
        if (value == 1 and (p not broadcast in earlier round))
            broadcast(p, 1, r);
            relay the r – 1 messages accepted in rounds 1, 2, …, r – 1
            that caused value to be set to 1;
        if (in rounds r’ ≤ r accepted (p_k, 1, r_k) from r distinct
            processes p_k including the transmitter p₁)
            value = 1;
    decide value;
Authenticated Arbitrary Consensus: II

- **Validity**
  - $m = 1$
    - $p_1$ sets value to 1 and broadcasts $(p_1, 1, 1)$.
    - By Correctness every correct process accepts $(p_1, 1, 1)$ in round 1 and sets value to 1.
    - value is never set to another value, so decides value.
  - $m = 0$
    - $p_1$ sets value to 0 and does not broadcast in round 1.
    - Since value = 0, $p_1$ never broadcasts in other rounds and decides on 0.
    - By Unforgeability no correct process ever accepts message from $p_1$.
    - Hence, no correct process sets value to 1, and so decides on 0.
Authenticated Arbitrary Consensus: III

- **Agreement**
  - Some correct $p$ first sets $value$ to 1 at the end of round $r < t + 1$:
    - $p$ accepted messages $(p_k, 1, r_k)$ from at least $r$ distinct processes $p_k$ including $p_1$.
    - In round $r + 1$ $p$ broadcasts $(p, 1, r + 1)$ and relays the $r - 1$ messages that caused $p$ to set $value$ to 1.
    - By Correctness and Relay, in round $r + 1$ all correct processes accept $(p, 1, r + 1)$ and $(p_k, 1, r_k)$ for $1 \leq k \leq r$, and so sets $value$ to 1.
  - Some correct $p$ first sets $value$ to 1 at the end of round $t + 1$:
    - $p$ accepted $(p_k, 1, r_k)$ from $t + 1$ distinct processes, and at least one must be correct. Say was $p_i$ broadcasting $(p_i, 1, r_i)$ where $r_i \leq t + 1$.
    - So $p_i$ set $value$ to 1 in round $r_i - 1 < t + 1$.
    - The first case above therefore holds.
  - Otherwise, each correct process has $value = 0$ and decides 0.
Simulating authenticated broadcast

- Implement a broadcast primitive that provides *Correctness*, *Unforgeability*, and *Relay* without using cryptographic methods

  - To broadcast a message, a set of processes need to witness the broadcast.
  - A correct process accepts a message only when it knows that there are sufficient witnesses to this broadcast.
  - Doing so prevents a faulty process from claiming to have received a message that was not sent to it, and to allow a correct process to prove why it accepted a message.
  - Requires $n > 3t$. 
Handy Facts about $n > 3t$

- Any subset of $n - 2t$ processes contains at least one correct process.
  \[ n - 2t > 3t - 2t = t. \]

- The intersection of any two subsets of $n - t$ processes contains at least one correct process.
  \[ n - t > 3t - t = 2t. \]

Example: $n = 7$, $t = 2$: $n - 2t = 3$, $n - t = 5$. 

\[ \begin{array}{cccccc}
  \text{Blue} & \text{Red} & \text{Blue} & \text{Red} & \text{Blue} & \text{Red} \\
\end{array} \]
Basic Idea

- Each round consists of two phases.
- $p$ sends $(init, p, m, k)$ to all.
- Any process that receives this message becomes a witness and sends $(echo, p, m, k)$.
- Any non-witness process that receives at least $n - 2t$ $(echo, p, m, k)$ also becomes a witness and sends $(echo, p, m, k)$.
  - Because in any set of $n - 2t$ processes, one must be correct.
- Any process that receives at least $n - t$ $(echo, p, m, k)$ accepts the message.
  - Because at least $n - 2t$ correct processes sent the echo, and so eventually all correct processes will echo.
Broadcast primitive

Round $k$:

*Phase $2k - 1$:* $p$ sends $(init, p, m, k)$ to all;

*Phase $2k$:* each process executes:
  - if received $(init, p, m, k)$ from $p$ in phase $2k - 1$
    - send $(echo, p, m, k)$ to all;
  - if received $(echo, p, m, k)$ from at least $n - t$ distinct processes in phase $2k$
    - accept $(p, m, k)$;

Round $r > k$:

*Phase $2r - 1, 2r$:* each process executes:
  - if received $(echo, p, m, k)$ from at least $n - 2t$ distinct processes in previous phases
    - and not sent $(echo, p, m, k)$
    - send $(echo, p, m, k)$ to all;
  - if received $(echo, p, m, k)$ from at least $n - t$ distinct processes in this and previous phases
    - accept $(p, m, k)$;
Correct process sends *init* in $2k - 1$

$n = 7, t = 2$
$n - t = 5$
$n - 2t = 3$
Correct processes accept in $2k$

$n = 7, t = 2$
$n - t = 5$
$n - 2t = 3$

also Unforgeability: if correct $p$ does not send $(init, p, m, k)$ then each correct process will receive no more than $t$ ($echo, p, m, k$) messages. Since $t < n - 2t$, there will be no more witnesses.
Faulty sends *init* in $2k - 1$

$n = 7, \ t = 2$
$n - t = 5$
$n - 2t = 3$
Correct can behave differently in $2k$: I

\[ n = 7, \; t = 2 \]
\[ n - t = 5 \]
\[ n - 2t = 3 \]
Correct can behave differently in $2k$: II

$n = 7, t = 2$
$n - t = 5$
$n - 2t = 3$
Correct can behave differently in $2k$: III

$n = 7, t = 2$
$n - t = 5$
$n - 2t = 3$
Relay: phase $i$

$n = 7, t = 2$
$n - t = 5$
$n - 2t = 3$
Relay: phase $i$

$n = 7, t = 2$
$n - t = 5$
$n - 2t = 3$
Relay: phase $i + 1$

$n = 7, t = 2$
$n - t = 5$
$n - 2t = 3$
Relay: phase $i + 1$

$n = 7, t = 2$
$n - t = 5$
$n - 2t = 3$
Delaying accept arbitrarily long: $2k$

$n = 7, \ t = 2$
$n - t = 5$
$n - 2t = 3$
Delaying accept arbitrarily long: $2k + 1$

$n = 7$, $t = 2$
$n - t = 5$
$n - 2t = 3$
Arbitrarily later... phase $i$

$n = 7$, $t = 2$
$n - t = 5$
$n - 2t = 3$
Arbitrarily later… phase $i + 1$

\begin{align*}
  n &= 7, \quad t = 2 \\
  n - t &= 5 \\
  n - 2t &= 3
\end{align*}
Arbitrarily later… phase $i + 2$

$n = 7$, $t = 2$
$n - t = 5$
$n - 2t = 3$
Nonauthenticated Arbitrary Consensus

// m ∈ {0, 1}, nonloquacious protocol

process p:
  if (p == p₁) value = m;
  else value = 0;
  for r =1 to t + 1
    if (value == 1 and (p not broadcast in earlier round))
      broadcast(p, 1, r);
    if (in rounds r' ≤ r accepted (pₖ, 1, rₖ) from r distinct processes pₖ including the transmitter p₁)
      value = 1;
  decide value;
Other results from this approach

- Multivalued agreement
- Voting
- Asynchronous randomized agreement
- Clock synchronization