ETH Zurich, Department of Computer Science Dr. Martin Hirt
SS 2017 Chen-Da Liu Zhang

Cryptographic Protocols
Notes 10

Scribe: Sandro Coretti (modified by Chen-Da Liu Zhang)

> *About the notes:* These notes serve as written reference for the topics not covered by the papers that are handed out during the lecture. The material contained therein is thus a *strict* subset of what is relevant for the final exam.

This week, the notes treat the important distributed primitives Broadcast and Consensus as well as protocols achieving these in the presence of an unbounded attacker corrupting up to $t < n/3$ of the parties. Moreover, they contain a proof that information-theoretically secure broadcast is not achievable if $t \geq n/3$.

10.1 Broadcast and Consensus: Definitions

This section presents Broadcast and Consensus. The former is a primitive that allows a certain player, called sender, to distribute a value to all players with the guarantee that all honest players receive the *same* value in the end. Moreover, if the sender is honest, then the players agree on the value sent by the sender. In Consensus, every player holds an input and the goal is that, in the end, the honest players agree on a value while preserving so-called *pre-agreement*.

More precisely, a *broadcast* protocol satisfies the following conditions:

- Consistency: All honest players output the same value, i.e., there is *agreement* at the end of the protocol.
- Validity: If the sender is honest, the value the honest players decide on is the value sent by him.
- Termination: All honest players decide on a value at some point.

Finally, a protocol achieves *consensus* if the follwing conditions are met:

- Consistency: All honest players output the same value, i.e., there is *agreement* at the end of the protocol.
- Persistency: If all honest players have the same input, they agree on this value.
- Termination: All honest players decide on a value at some point.

Note that, if $t < n/2$, broadcast and consensus are equivalent in the sense that a broadcast protocol can easily be transformed into a consensus protocol and vice-versa:

Broadcast $\Rightarrow$ Consensus: Each player broadcasts his value and decides on the value received most often.

Consensus ⇒ Broadcast: The sender sends the value to be broadcast to all players. Then, the players run consensus, where each player inputs the value received from the sender.

In the following we present the consensus protocol by [BGP89], which provides information-theoretic security provided that less than a third of the players are corrupted, i.e., $t < n/3$. We start out with a protocol that achieves a weak form of consensus and then build from it a full consensus protocol, which can be transformed into a broadcast protocol as outlined above.

The protocol by [BGP89] is a binary protocol, i.e., a protocol with input space $\{0, 1\}$. A protocol with a larger input space can, e.g., be achieved by parallel execution of the one-bit protocol.

10.2 Constructing Consensus

10.2.1 Weak Consensus

In Weak Consensus, each player has an input x_i and eventually decides on a value $y_i \in \{0, 1, \perp\}$, where $\perp$ is to be considered "invalid." Weak Consensus achieves a weaker consistency condition:

Weak Consistency: If some honest player P_i decides on an output $y_i \in \{0, 1\}$, then $y_j \in \{y_i, \perp\}$ for all honest players P_j.

In other words, no two honest players decide on two different "valid" outputs. A protocol achieves Weak Consensus if it satisfies weak consistency, persistency, and termination. The following is such a protocol:

```
Protocol WeakConsensus $(x_1, \ldots, x_n) \to (y_1, \ldots, y_n)$:
  1. $\forall P_i$: send $x_i$ to each $P_j$
  2. $\forall P_j$: $y_j = \begin{cases} 0 & \text{if number of received zeros} \geq n - t \\ 1 & \text{if number of received ones} \geq n - t \\ \perp & \text{otherwise} \end{cases}$
  3. $\forall P_j$: return $y_j$
```

Lemma 10.1. *If $t < n/3$, protocol* `WeakConsensus` *achieves persistency, weak consistency, and termination.*

Proof. Persistency: If all honest players input the same value x, each honest player receives x at least $n - t$ times (and at most $t < n - t$ times the value $1 - x$) and thus decides on x.

Weak Consistency: Suppose an honest player P_i decides on y_i. Then, he received the value y_i at least $n - t$ times in step 1. Since at least $n - 2t$ of these messages are from honest players, it follows that every honest player has received y_i at least $n - 2t$ times and thus $1 - y_i$ at most $2t < n - t$ times. Hence, no honest player P_j decides $y_j = 1 - y_i$.

Termination: Obvious. □

10.2.2 Graded Consensus

Each player starts out with an input $x_i \in \{0, 1\}$ and eventually decides on a value $y_i \in \{0, 1\}$ and on a grade $g_i \in \{0, 1\}$. The grade $g_i = 1$ is to be considered as "consistency achieved," whereas $g_i = 0$ means "not sure consistency is achieved."

We introduce the following requirements:

GRADED CONSISTENCY: If some player P_i decides on an output $y_i \in \{0,1\}$ with grade $g_i = 1$, then $y_j = y_i$ for all honest players P_j.[1]

GRADED PERSISTENCY: If all honest players P_i have the same input x, they all decide on the output $(y_i, g_i) = (x, 1)$.

A protocol achieves Graded Consensus if it satisfies graded consistency, graded persistency, and termination. Such a protocol can be constructed as follows:

Protocol `GradedConsensus` $(x_1, \ldots, x_n) \to ((y_1, g_1), \ldots, (y_n, g_n))$**:**

```
1. $(z_1, \ldots, z_n) = \texttt{WeakConsensus}(x_1, \ldots, x_n)$
2. $\forall P_i$: send $z_i$ to each $P_j$.
3. $\forall P_j$
4. $\forall P_j$: return $(y_j, g_j)$
```

$$y_j = \begin{cases} 0 & \text{if \#zeros} \geq \text{\#ones} \\ 1 & \text{if \#zeros} < \text{\#ones} \end{cases}$$

$$g_j = \begin{cases} 1 & \text{if } \#y_j\text{'s} \geq n - t \\ 0 & \text{otherwise} \end{cases}$$

Lemma 10.2. *If $t < n/3$, protocol* `GradedConsensus` *achieves graded persistency, graded consistency, and termination.*

Proof. GRADED PERSISTENCY: Assume all honest players input the same value x. The persistency of `WeakConsensus` guarantees that all honest players send x in step 2. Thus, each honest player P_i receives x at least $n - t$ times (and $1 - x$ at most t times) and thus decides on $y_i = x$ und $g_i = 1$.

GRADED CONSISTENCY: Suppose honest player P_i outputs $g_i = 1$. Then, he has received y_i from at least $n - t$ players in step 2. Therefore, any other honest player P_j has received y_i at least $n - 2t$ times. Furthermore, since $n - 2t > t$, at least one honest player obtained y_i as output of `WeakConsensus`. Therefore, by WEAK CONSISTENCY, no honest player output $1 - y_i$ in `WeakConsensus`, from which it follows that P_j received $1 - y_i$ at most $t < n - 2t$ times and therefore outputs $y_j = y_i$.

TERMINATION: Obvious. □

10.2.3 King Consensus

In King Consensus some player P_k takes over the role of the king. If the king is honest, we require that the protocol achieve consensus. If the king is corrupted, at least pre-agreement should be preserved.

Thus, we introduce the following new requirement:

KING CONSISTENCY: If the king P_k is honest, all players decide on the same value in the end.

A protocol achieves King Consensus if it satisfies king consistency, persistency, and termination. Such a protocol can be constructed as follows:

[1] But not necessarily $g_j = 1$.

```
Protocol  $\texttt{KingConsensus}_{P_k}(x_1, \ldots, x_n) \rightarrow (y_1, \ldots, y_n)$:
  1. $((z_1, g_1), \ldots, (z_n, g_n)) = \texttt{GradedConsensus}(x_1, \ldots, x_n)$
  2. $P_k$: send $z_k$ to each player $P_j$.
  3. $\forall P_j$:   $y_j = \begin{cases} z_j & \text{if } g_j = 1 \\ z_k & \text{otherwise} \end{cases}$
  4. $\forall P_j$: return $y_j$
```

Lemma 10.3. *If $t < n/3$, protocol* `KingConsensus` *achieves persistency, king consistency, and termination.*

Proof. Persistency: If all honest players P_i start the protocol with the same input x, then graded persistency implies that after step 1, they hold $(z_i, g_i) = (x, 1)$ and decide $y_i = z_i = x$ in step 3.

King Consistency: Suppose the king is honest. If all honest players P_i have $g_i = 0$ in step 1, then they all take king's value. Otherwise, let P_j be a player with $g_j = 1$. Graded consistency implies that in such a case all honest players P_i have $z_i = z_j$. In particular, this holds for the (honest) king P_k. Thus, all players decide on the same output.

Termination: Obvious. □

10.2.4 Consensus

Consensus can be achieved from King Consensus by running `KingConsensus` with $t+1$ different kings P_k, which guarantees that at least one of them is honest.

```
Protocol  $\texttt{Consensus}(x_1, \ldots, x_n) \rightarrow (y_1, \ldots, y_n)$:
  1. for $k := 1$ to $t+1$ do
             $(x_1, \ldots, x_n) := \texttt{KingConsensus}_{P_k}(x_1, \ldots, x_n)$
     od
  2. $\forall P_j$: return $y_j = x_j$
```

Lemma 10.4. *Protocol* `Consensus` *achieves Consensus if at most $t < n/3$ players are corrupted.*

Proof. Persistency: Assume there is pre-agreement at the beginning of the protocol. Then, the consistency of `KingConsensus` (Lemma 10.3) immediately implies that it is preserved until the end.

Consistency: At least one of the kings $P_k \in \{P_1, \ldots, P_{t+1}\}$ is honest. Thus, after the execution of $\texttt{KingConsensus}_{P_k}$, there is agreement among the honest parties, which is maintained until the end of the protocol due to the persistency condition.

Termination: Obvious. □

10.3 Impossibility of Consensus for $t \geq n/3$

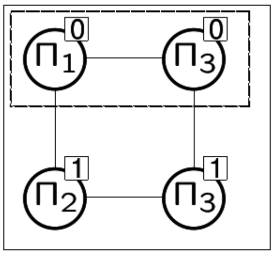
Similarly to previous impossibility proofs (cf. Section 7.3 and 9.1), we first consider the setting $t = 1$ and $n = 3$. Then, the general case follows analogously (and is not repeated here).

Towards a contradiction, suppose there exists a Consensus protocol for three parties tolerating one corruption. Such a protocol is given by three deterministic[2] protocol machines Π_1, Π_2, and

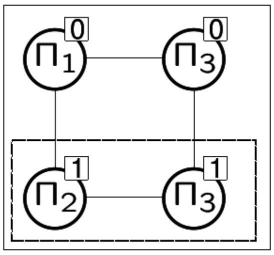
[2] The argument can be extended to randomized machines.

Π_3 for parties P_1, P_2, and P_3, respectively, where each machine expects to be connected to two other machines.

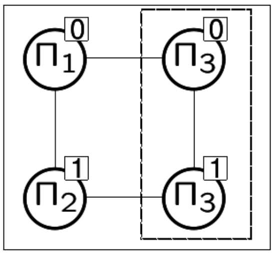
Assume an attacker corrupts P_1 in a normal execution of the protocol and emulates Π_1 and Π_3 as shown below in Case 1, where the number in the square next to a protocol machine denotes the machine's input. The Persistency property of Consensus implies that machine Π_2 of P_2 (in the bottom left corner) must output 1, since the honest players have pre-agreement.



Case 1



Case 2

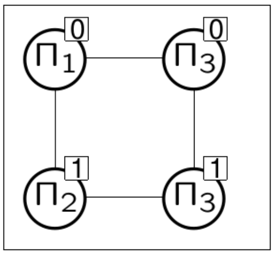


Case 3

If the attacker corrupts P_2 and emulates Π_2 and Π_3 as shown in Case 2, then Persistency implies that machine Π_1 of P_1 (in the top left corner) must output 0, since the honest players have pre-agreement.

Finally, suppose the attacker corrupts P_3 and emulates two copies of Π_3 as shown in Case 3. In that case, by Consistency, Π_1 and Π_2 (on the left-hand side) must output *equal* values.

Note that all three cases are actually one and the same setup:



However, the conditions derived above constitute a contradiction: It is impossible that the output of Π_0 is 0, that of Π_1 is 1, and both of them are equal simultaneously. Thus, there exists no secure consensus protocol for $t = 1$ and $n = 3$.

References

[BGP89] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *FOCS*, pages 410–415, 1989.