Resolve-impossibility for a contract signing protocol

Aybek Mukhamedov and Mark Ryan

July 6, 2006
Outline

1. Multi-party contract signing
2. A protocol by Garay and MacKenzie
3. A revised protocol by Chadha, Kremer and Scedrov
4. A flaw in the revised protocol
5. Impossible to “resolve”
Digital Contract Signing

- Use digital signatures to sign a pre-agreed contract over a computer network
- Potentially useful for e-commerce
- Why it is not simple:

  \[
  A \rightarrow B : \text{Sign}_A(\text{contract}) \\
  B \rightarrow A : \text{Sign}_B(\text{contract})
  \]

Someone has to start first.
**Contract Signing protocol**

- Main property: **fairness**
  - 2-party: if A gets B’s signature, then B can get A’s signature, and vice-versa
  - $n$-party: if any agent gets a signature from any other agent, then all agents can get signatures from every other agent.

- Must not fail in the presence of a Dolev-Yao attacker on the network

- ... controlling a coalition of up to $n - 1$ dishonest agents
Solutions

- Use trusted party $T$ to collect and distribute the signed contracts
  - Problem: $T$ may become a bottleneck.
- **Optimistic** protocols:
  - The agents can complete the contract signing without $T$ (optimistic case)
  - $T$ will be invoked and will take decisions iff something goes amiss.
  - Channels between parties and $T$ are resilient.
"Optimistic" protocols: 2-party

- $T$ will enforce the contract if presented with both promises
- More involved for $n$-party
“Optimistic” protocols: $T$

- $T$ can **enforce the contract** by converting promises to signatures
  - it will do so if it has proof that all parties have issued a promise
- $T$ can issue an **abort token**
  - 2-party: means that it will not enforce contract
  - n-party: means that it will not enforce contract; but it may overturn this abort decision if presented with evidence of cheating by the signer that got the abort
- $T$ acts only when requested by an agent
  - decides whether to abort or resolve based on the evidence in the complaint
“Optimistic” protocols

- Optimistic synchronous multi-party contract signing:
  - Asokan, Baum-Waidner, Schunter, Waidner, 1998

- Optimistic asynchronous multi-party contract signing:
  - Baum-Waidner, Waidner, ICALP 2000 and 2001
  - Garay, MacKenzie, DISC 1999;
Garay-MacKenzie protocol

- Two parts:
  - Main protocol: defines actions for signers
  - Resolve protocol: defines actions for a $T$

- Signers’ promises are private contract signatures (Garay, et al [CRYPTO’99]):
  - $PCS_A(m, B, T)$ is a promise from $A$ to $B$ on $m$
  - Only $B$ and $T$ can verify its validity
  - $T$ can convert it into a conventional digital signature that binds $A$ on $m$
GM: main protocol

- Signers: $P_1, \ldots, P_n$
- The protocol is divided into $n$ levels:
  - Promises are level-specific, i.e. they are of the form $PCS_A((m, i), B, T)$, where $i = 0, \ldots, n + 1$
  - The $i$th-level is triggered when $P_i$ receives 1st-level promises from $P_{i+1}$ through $P_n$
  - In the $i$th-level signers $P_i$ through $P_1$ exchange $i$th-level promises
  - $P_i$ through $P_1$ close higher levels
- After the $n$th-level actual signatures are exchanged
GM: main protocol

\[ P_i \quad P_{i-1} \quad \ldots \quad P_1 \]

- Distribute 1-level promises to \( P_{<i} \)
- \( i \) – 1-level protocol
- Collect \( i \) – 1-level promises
- Exchange \( i \)-level promises
Depending on the level of the protocol execution a signer $P_i$ may:
- Quit the protocol $P_i$ if did not send any promises
- Request $T$ to intervene

Each signer may contact $T$ only once

$T$ replies with a resolved contract or an abort token

$T$ may overturn its abort decision, but never resolve
GM: resolve protocol

- The resolve protocol defines what $T$ replies to signers’ requests
- Found to be flawed by Chadha, Kremer and Scedrov (CSFW 2004): attacks on fairness involving four (and more) signers
- Proposed a revised resolve protocol:
  - Abort is overturned iff $T$ infers that each signer that contacted it in the past has been dishonest
- Verified with model-checker MOCHA for protocol runs involving three and four signers
CKS: resolve protocol

- $P_i$ requests recovery with:

$$S_{P_i} \left( \{ PCS_{P_j}((m, \tau_j), P_i, T) \}_{j \in \{1, \ldots, n\} \setminus \{i\}}, S_{P_i}((m, 1)) \right)$$

where $\tau_j$ is the (appropriate) level of promise from $P_j$ to $P_i$.

- $T$ stores names of agents in a set $S(m)$ to whom it has replied with abort.

- For each $P_i$ in $S(m)$, $T$ deduces the highest level promises $P_i$ could have sent to higher and lower indexed agents:
  - $T$ infers $P_i$’s dishonest iff it is later presented with a higher level promise issued by $P_i$. 
The revised protocol is still flawed – attacks on fairness involving five signers:

- $P_1, \ldots, P_5$ optimistically execute the protocol until $P_4$ sends out its signature on a contract $m$.
- $P_1$, $P_2$ and $P_3$ do not send their signatures to $P_4$.
- $P_5$ requests abort and $P_3, P_2, P_1$ request resolve from $T$.
- $P_4$ requests resolve from $T$, but gets abort.
Our analysis: five signers
The attack applies to runs with any $n > 4$ signers:
- $P_1, \ldots, P_n$ optimistically execute the protocol until $P_4$ sends out its signature on a contract $m$.
- $P_1$ and $P_3$ do not send their signatures to $P_4$.
- $P_n$ requests abort and $P_3, P_2, P_1$ request resolve from $T$.
- $P_4$ requests resolve from $T$, but gets abort.

Idea of the attacks: a coalition of dishonest signers propagates $T$’s abort decision
Our analysis: more signers
Our analysis: resolve impossibility

- Attacks do not depend on the resolve protocol:
  - for any resolve protocol, the main protocol is subject to attacks on fairness
- Resolve impossibility follows from case-by-case analysis of $T$’s actions in the previous attack:
  - no matter what $T$ does, it is unfair to someone, who could be honest.
If $P_n$ requests abort claiming not to have received dotted messages, $T$ must grant it.
Our analysis: resolve impossibility

If $P_1$ requests resolve, $T$ must confirm previous abort.
Our analysis: resolve impossibility

If $P_3$ requests resolve, $T$ must still confirm previous abort
Our analysis: resolve impossibility

If $P_2$ requests resolve, $T$ must still confirm previous abort
Our analysis: resolve impossibility
Conclusion

- Garay and MacKenzie protocol broken and fixed by Chadha, Kremer and Scedrov: the new protocol was verified for runs with three and four signers.
- New attack on the fixed protocol involving $n > 4$ signers.
- Our attack also shows that the idea behind the main protocol does not work – no resolve protocol will fix it.

Future work

- New protocol preserving the ideas of Garay/Mackenzie and Chadha/Kremer/Scedrov:
  - Private contract signatures (abuse-freeness for free)
  - Cascading promises
  - Elegant procedure for resolve protocol