On Security Models and Compilers for Group Key Exchange Protocols

Emmanuel Bresson  
DCSSI Crypto Lab  
Paris, France

Mark Manulis  
UCL Crypto Group  
Louvain-la-Neuve, Belgium

Jörg Schwenk  
Horst Götz Institute for IT Security  
Bochum, Germany

Group Key Exchange (GKE)

[Diagram of Group Key Exchange with users and a communication channel]
Core Security Requirements for GKE

- Authenticated Key Exchange (AKE)
  - indistinguishability of the key from a random
  - known-key security
  - forward secrecy (corruptions in the future should keep secrecy of session keys in the past)

- Mutual Authentication (MA)
  - assurance of participation
  - key confirmation

Security Models for GKE

- needed to provide *provable security*

- model protocol execution
  - participants
  - sessions (concurrent/sequential executions)

- model security requirements
  - adversarial capabilities
  - adversarial goals (games)
Participants

- \( U \) is a set of \( n \) users in the universe
- every \( U_i \in U \) has a long-lived key \( LL_i \)
- not enough for concurrent sessions

Sessions

- unlimited number of instances \( \Pi(i,s) \) for \( U_i \)
- every instance becomes associated with
  - session group key \( k(i,s) \)
  - session id \( sid(i,s) \)
  - partner id \( pid(i,s) \)

how to identify instances of the same session?
Matching Conversations \([BR93]\)

\[ \Pi(1,s) \quad \Pi(2,s) \]

- existence of matching conversations ensures
  - participation of the instances in the execution, thus instances are \textit{partnered}
  - acceptance of the instances with identical keys
- technical core of MA-security in \([BR93]\)

Matching Conversations in GKE?

- MCs are not sufficient for all GKE (i.e., only for broadcast GKEs as in \([KY03]\))

\[ \Pi(1,s) \quad \Pi(2,s) \quad \Pi(3,s) \]

- attempt to extend MCs to groups in \([BCPQ01]\)
- definition of the \textit{partnering condition} through \textit{partially} matching conversations
Session IDs in BCPQ

\[
\begin{align*}
\Pi(1,s) & \xrightarrow{m_1} \text{comm. channel} & m_2 & \xrightarrow{m_3} \Pi(2,s) & \xrightarrow{m_3} \Pi(3,s)
\end{align*}
\]

- session IDs are built from the union of partial conversations

<table>
<thead>
<tr>
<th>sid(i,s)</th>
<th>sid_{i1}</th>
<th>sid_{i2}</th>
<th>sid_{i3}</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid(1,s)</td>
<td>-</td>
<td>m_1</td>
<td>m_3</td>
</tr>
<tr>
<td>sid(2,s)</td>
<td>m_1</td>
<td>-</td>
<td>m_2 \mid m_3</td>
</tr>
<tr>
<td>sid(3,s)</td>
<td>m_3</td>
<td>m_2 \mid m_3</td>
<td>-</td>
</tr>
</tbody>
</table>

Partnering Condition in BCPQ

- \(\Pi(i,s)\) and \(\Pi(j,s)\) are *directly partnered* if
  - they have accepted and
  - \(\text{sid}(i,s) \cap \text{sid}(j,t) \neq \emptyset\)
  - denoted \(\Pi(i,s) \leftrightarrow \Pi(j,t)\)

- \(\Pi(i,s)\) and \(\Pi(j,s)\) are *partnered* if
  - there is a sequence \(\Pi(l_1,s), \ldots, \Pi(l_t,s)\)
  - \(\Pi(l_1,s) = \Pi(i,s), \Pi(l_t,s) = \Pi(j,s)\)
  - \(\Pi(l_{i-1},s) \leftrightarrow \Pi(l_i,s)\) for all \(|l| = l_2, \ldots, l_t\)
Example of Partnering in BCPQ

\[ \Pi(1, s) \xrightarrow{m_1} \Pi(2, s) \xrightarrow{m_2} \Pi(3, s) \]

<table>
<thead>
<tr>
<th>sid(1, s)</th>
<th>sid(2)</th>
<th>sid(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid(1, s)</td>
<td>-</td>
<td>( m_1 )</td>
</tr>
<tr>
<td>sid(2, s)</td>
<td>( m_1 )</td>
<td>-</td>
</tr>
<tr>
<td>sid(3, s)</td>
<td>( m_3 )</td>
<td>( m_2 \mid m_3 )</td>
</tr>
</tbody>
</table>

- \( \Pi(1, s) \leftrightarrow \Pi(2, s) \) since sid\(_{12} = \) sid\(_{21} \)
- \( \Pi(2, s) \leftrightarrow \Pi(3, s) \) since sid\(_{23} = \) sid\(_{32} \)
- \( \Pi(3, s) \leftrightarrow \Pi(1, s) \) since sid\(_{31} = \) sid\(_{13} \)

Drawback of BCPQ Partnering

\[ \Pi(1, s) \xrightarrow{m_1} \Pi(2, s) \xrightarrow{m_2} \Pi(3, s) \]

<table>
<thead>
<tr>
<th>sid(1, s)</th>
<th>sid(2)</th>
<th>sid(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid(1, s)</td>
<td>-</td>
<td>( m_1 )</td>
</tr>
<tr>
<td>sid(2, s)</td>
<td>( m'_1 )</td>
<td>-</td>
</tr>
<tr>
<td>sid(3, s)</td>
<td>( m_3 )</td>
<td>( m_2 \mid m_3 )</td>
</tr>
</tbody>
</table>

- \( \Pi(1, s) \leftrightarrow \Pi(2, s) \) since sid\(_{12} \neq \) sid\(_{21} \), but
- \( \Pi(1, s) \leftrightarrow \Pi(3, s) \) and \( \Pi(3, s) \leftrightarrow \Pi(2, s) \), thus
- \( \Pi(1, s) \) and \( \Pi(2, s) \) are still partnered

Insufficient for MA-security!
A More General Partnering Condition

- \( \Pi(i,s) \) and \( \Pi(j,s) \) are **partnered** if
  - they have accepted
  - \( \text{sid}(i,s) = \text{sid}(j,s) \)
  - \( \text{pid}(i,s) = \text{pid}(j,s) \) where \( \text{pid}(i,s) \) consists of all \( U_j \in \mathcal{U} \) participating in the session

- used e.g. in [KS05] for UC-secure GKE where \( \text{sid}(i,s) \) is given as input

- we assume, either \( \text{sid}(i,s) \) is given as input or it is defined in the GKE protocol

---

Adv. Queries to \( \Pi(i,s) \) \([BCPQ01,KY03,BCP02]\)

**Passive/Active Attacks**
- Send(\( \Pi(i,s), m \)) sends \( m \) to \( \Pi(i,s) \)

**Known-Key Attacks**
- RevealKey(\( \Pi(i,s) \)) returns \( k(i,s) \), if accepted

**Corruptions**
- next slide
Weak vs. Strong Corruptions

Current Separation in GKE

- **weak corruption model**[^BCPQ01, KY03]
  - access to Corrupt(U_i) which reveals LL_i
- **strong corruption model**[^BCP02]
  - access to Corrupt(Π(i,s)) which reveals LL_i and state(i,s)
  - impossible to reveal state(i,s) without LL_i

Separation by [CK01] for 2-Party-KE

- [CK01] specifies two queries
  - Corrupt(U_i) reveals LL_i
  - RevealState(Π(i,s)) reveals state(i,s)

Current Separation in 2-Party-KE

- **weak corruption model**
  - access to Corrupt(U_i) but not to RevealState(Π(i,s))
- **strong corruption model**
  - access to Corrupt(U_i) and to RevealState(Π(i,s))
  - possible to ask RevealState(Π(i,s)) w/o Corrupt(U_i)

- technical core of AKE-Security
- adversary tries to distinguish keys accepted by fresh instances from some random values
- $\Pi(i,s)$ is fresh if
  - it has accepted with some $k(i,s)$
  - no $\text{RevealKey}(\cdot)$ to $\Pi(i,s)$ or to any partner $\Pi(j,s)$
  - to model (strong) forward secrecy \cite{BCP01,KY03,BCP02}
  - no $\text{Corrupt}(\Pi(i,s))$ before $\text{Send}(\Pi(j,s), m)$

---

Our Refinements: Corruption Models

- based on separation of queries \cite{CK01}
- in addition to $\text{Send}$ and $\text{RevealKey}$ queries

<table>
<thead>
<tr>
<th></th>
<th>weak corruption model (wcm)</th>
<th>wcm forward secrecy (wcm-fs)</th>
<th>wcm backward secrecy (wcm-bs)</th>
<th>strong corruption model (scm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrupt($U_i$)</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>RevealState($\Pi(i,s)$)</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

- wcm equiv. to secrecy w/o corruptions
- wcm-fs equiv. to weak corruptions \cite{BCP01,KY03,CK01}
- scm equiv. to strong corruptions \cite{BCP02,CK01}
Our Refinements:
Secrecy Types / Instance Freshness

- \textbf{wfs} equiv. to forward secrecy\textsuperscript{[BCPQ01,KY03]}
  - no Corrupt(\Pi(i,s)) before Send(\Pi(j,s), m)

- \textbf{sfs} equiv. to strong forward secrecy\textsuperscript{[BCP02]}
  - as in wfs and no RevealState to \Pi(i,s) or its partners before they accept

- \textbf{wbs} (weak backward secrecy)
  - no RevealState to \Pi(i,s) or its partners after protocol invocation

- \textbf{sbs} (strong backward secrecy) \textit{(of theoretical value)}
  - as in wbs and no Corrupt(\Pi(i,s)) before Send(\Pi(j,s), m) after protocol invocation

Security Compiler C\textsubscript{AMA} \textsuperscript{[KY83] + [KS95]}

\textbf{User 1} \quad \textbf{User 2}

\begin{align*}
\text{choose random } r_1 \\
\text{sid}_1 = r_1 | r_1
\end{align*}

\begin{align*}
\text{computation of the unique session id}
\end{align*}

\begin{align*}
\text{m} \quad \text{sign} = \text{Sign} (sk_1, m | \text{sid}_1 | \text{pid}_1) \quad \text{m} | \sigma_1
\end{align*}

\begin{align*}
\text{Verify} (pk_1, m | \text{sid}_2 | \text{pid}_2, \sigma_1) \quad \text{P'}
\end{align*}

\begin{align*}
\sigma_1 = \text{PRF} (k_1, y_1) \\
\sigma_1 = \text{Sign} (sk_1, \mu_1 | \text{pid}_1 | \text{sid}_1) \\
\text{Verify} (pk_1, \mu_1 | \text{sid}_1 | \text{sid}_1, \sigma_1) \\
K_1 = \text{PRF} (k_1, y_2) \\
\text{erase state and accept with } K_1
\end{align*}

\begin{align*}
\sigma_2 = \text{PRF} (k_2, y_1) \\
\sigma_2 = \text{Sign} (sk_2, \mu_2 | \text{pid}_2 | \text{sid}_2) \\
\text{Verify} (pk_2, \mu_2 | \text{sid}_2 | \text{sid}_2, \sigma_2) \\
K_2 = \text{PRF} (k_2, y_2) \\
\text{erase state and accept with } K_2
\end{align*}
Security of $C_{\text{AMA}}$

- we show that $C_{\text{AMA}}$ preserves AKE-Security for
  - weak/strong forward secrecy, weak backward secrecy
  - original [KY03] considered only weak forward secrecy
  - [KY03] provides stronger security than originally proven, but
  - original [KY03] is not generic (more on this in the paper)

provides MA-Security for

- scm
- implies security of [KS05] even if the adversary reveals state($\Pi(i,s)$) for instances of uncorrupted $U_i$
- possible due to the separation of queries
- [KS05] provides stronger security than originally proven

Summary of Contributions

- identify weaknesses of partnering in [BCPQ01]
- suggest more general treatment of partnering
- adapt separation of Corrupt and RevealState$^{[\text{CK01}]}$ to GKE
- refine AKE with weak/strong backward/forward secrecy
- redefine MA while allowing RevealState to uncorrupted users (not in the talk)
- define $C_{\text{AMA}}$ as mod. combination of [KY03] and [KS05]
- show why original [KY03] is not generic (not in the talk)
- prove that $C_{\text{AMA}}$ satisfies our AKE and MA (not in the talk)