Securing Group Key Exchange against Strong Corruptions

Emmanuel Bresson
DCSSI Crypto Lab
Paris, France

Mark Manulis
UCL Crypto Group
LLN, Belgium

ASIACCS 2008, 18-20 March, Tokyo, Japan
Group Key Exchange (GKE)

long-lived keys for authentication (\(sk_i, pk_i\)) or \(pw_i\)

secret contributions/ephemeral secrets/internal states

session group key

state_i

communication channel
Security Requirements for GKE

- **Authenticated Key Exchange (AKE)** \([\text{BCPQ01, BCP02, KY03}]\)
  - indistinguishability of the key from a random
  - known-key security (keys from other sessions leak)
  - forward secrecy (leakage of LLs in the future)

- **Mutual Authentication (MA)** \([\text{BCPQ01, BMS07}]\)
  - assurance of participation
  - prevention of unknown key share attacks
  - key confirmation/agreement

- **Contributiveness** \([\text{AST98, BM07}]\)
  - unpredictability of the group key, no enforcement of the key
  - prevention of key replication attacks \([\text{K05}]\)
Session Parameters

• unlimited number of instances $\Pi(i,s)$ for $U_i$

• every instance is associated with
  – partner id $\text{pid}(i,s) = U_1|...|U_n$
  – session id $\text{sid}(i,s)$
  – internal state/ephemeral secrets $\text{state}(i,s)$
  – session group key $k(i,s)$

• session instances share sid and pid $^{[KS05,BMS07]}$
Modeling Attacks via Queries

• passive/active attacks
  – Execute($U_1, \ldots, U_n$) gives execution transcript
  – Send($\Pi(i,s), m$) sends $m$ to $\Pi(i,s)$ and receives the output message of $\Pi(i,s)$

• leakage of session keys
  – RevealKey($\Pi(i,s)$) returns $k(i,s)$
  – answered if $\Pi(i,s)$ has successfully completed the session

• weak/strong corruptions
Corruptions in GKE

- weak corruption model in GKE\textsuperscript{[BCPQ01,KY03]}
  \text{Corrupt}(U_i) \text{ reveals } LL_i

- strong corruption model in GKE\textsuperscript{[BCP02,KS05]}
  \text{Corrupt}(\Pi(i,s)) \text{ reveals } LL_i \text{ and } \text{state}(i,s)

What about revealing \text{state}(i,s) without \text{LL}_i?
Look in 2-Party Key Exchange

- 2 queries: $\text{Corrupt}(U_i)$ and $\text{RevealState}(\Pi(i,s))$ [CK01]

- weak corruption model in 2-KE
  $\text{Corrupt}(U_i)$ but not $\text{RevealState}(\Pi(i,s))$

- strong corruption model in 2-KE
  $\text{Corrupt}(U_i)$ and $\text{RevealState}(\Pi(i,s))$

allows $\text{RevealState}(\Pi(i,s))$ without $\text{Corrupt}(U_i)$
Opening Attacks

- adversary opens $\Pi(i,s)$ through $\text{RevealState}(\Pi(i,s))$  
  $\rightarrow$ $\Pi(i,s)$ remains honest

- adversary corrupts $U_i$ through $\text{Corrupt}(U_i)$  
  $\rightarrow$ all $\Pi(i,s)$ become malicious

- opening attacks make instances transparent  
  all ephemeral secrets used to derive $k(i,s)$ become visible

- can be used for finer and stronger requirements/models
Stronger Security Requirements

• Strong AKE-security
  – no malicious participants in attacked session
  – no opened participants in attacked session
  – opening attacks before the session starts and after it finishes

• Strong MA-security
  – at least 2 session participants honest (uncorrupted)
  – up to n-2 participants malicious (corrupted)
  – no restrictions on the number of opened participants

• Strong Contributiveness
  – at least 1 participant honest (uncorrupted)
  – up to n-1 participants malicious (corrupted)
  – no restrictions on the number of opened participants
Tree Diffie-Hellman Keys

- used in GKE protocols [SSDW88,BW96,P99,KPT00,KPT01]
Requirements on $G = \langle g \rangle$

- cyclic group $G = \langle g \rangle$ of prime order $q$
- DDH is believed to be hard in $G$

$x$ is uniform and random in $\mathbb{Z}_q$

$g^x$ is uniform and random in $\mathbb{Z}_q$

**Solution:**

- efficient bijective mapping from $\mathbb{Z}_q$ to $G$, and
- efficient bijective mapping from $G$ to $\mathbb{Z}_q$ (not log!)
Suitable $G=\langle g \rangle^{[KPT00]}$

- $p = 2q+1$ for a large prime $q$
- Let $\hat{G} = \langle g \rangle$ be $\text{QR}(p)$ //quadratic residues mod $p$
- mapping $u : \hat{G} \rightarrow \mathbb{Z}_q$

$$u(z) = \begin{cases} 
  z \mod q & \text{if } z \leq q \\
  p - z \mod q & \text{if } q < z < p
\end{cases}$$

- Let $G = \{u(g^i) \mid i \in \mathbb{Z}_q\}$. It can be shown that $G = \mathbb{Z}_q$

- $g^x$ defined as $u(g^x)$ is a bijection from $\mathbb{Z}_q$ to $G$
Tree Decisional Diffie-Hellman

- **TDDH problem**
  
  For any *full* binary tree with \( n \) leaves, any suitable \( G \)

\[
g^{x_{1,0}x_{1,1}} \approx g^r
\]

- **TDDH \iff DDH**
  
  \[
  \text{Adv}^{\text{DDH}} \leq \text{Adv}^{\text{TDDH}} \leq (2n-3)\text{Adv}^{\text{DDH}}
  \]
TDH1 Key Structure

- unauthenticated GKE by Kim-Perrig-Tsudik 2001

- Round 1
  choose random $x_{i,v_i}$
  $U_i$ broadcasts $y_{i,v_i} = g^{x_{i,v_i}}$

- Round 2
  $U_1$ at position $<n-1,0>$ broadcasts
  $Y = \{ g^{x_{n-2,0}}, \ldots, g^{x_{1,0}} \}$

- Key Computation
  $U_i$ computes $X = \{ x_{i-1,0}, \ldots, x_{0,0} \}$
  key material is $x_{0,0}$
Adding Authentication

- almost based on the compiler by Katz-Yung 2003

- Round 1
  - $U_i$ chooses nonce $r_i$
  - computes $\sigma_i$ on $0|y_{i,v_i}|r_i|\text{pid}_i$
  - broadcasts $0|y_{i,v_i}|r_i|\sigma_i$

- Round 2
  - $U_i$ verifies all $\sigma_j$
  - computes $\text{sid}_i = r_1|...|r_n$
  - $U_1$ computes $\sigma'_1$ on $1|Y|\text{sid}_1|\text{pid}_1$
  - broadcasts $1|Y|\sigma'_1$

- Key Computation
  - $U_1$ verifies $\sigma'_1$ and computes $x_{0,0}$
Replay Attack against AKE

session A

$\sigma'_3$

$y_{1,0}$

$r_3$

$y_{2,0}$

$\sigma_3$

$U_3$

$\text{sk}_3$

$\text{pk}_3$

$\text{sk}_4$

$\text{pk}_4$

$x_{0,0}$

$U_4$

$y_{1,1}$

$r_6$

$\sigma_6$

$U_6$

$\text{sk}_6$

$\text{pk}_6$

session B

$\sigma'_2$

$y_{1,0}$

$r_5$

$y_{2,0}$

$\sigma_2$

$U_5$

$\text{sk}_5$

$\text{pk}_5$

$\text{sk}_4$

$\text{pk}_4$

$x_{0,0}$

$y_{1,1}$

$U_4$

$y_{2,1}$

$r_4$

$\sigma_4$

$U_2$

$\text{sk}_2$

$\text{pk}_2$

attack is feasible due to interleaved nonces
Opening Attack against AKE

- all users accept → session is finished
- asks RevealState(U_i)
- learns x_{i,v_i}, x_{i-1,0} ....
- can recompute x_{0,0}

attack against strong forward secrecy
Replay Attack against MA

unknown key-share attack
Malicious Participants against MA

- Malicious U sends $y_{1,0}$ to $U_2$
- Malicious U sends $y_{1,0}$ to $U_3$

- $U_2$ computes $x_{0,0}$
- $U_3$ computes $x_{0,0}$

No agreement on $x_{0,0}$ between $U_2$ and $U_3$
Collusion against Contributiveness

\[ g^{x_{2,0}x_{2,1}} = x_{1,0} \]

\[ x_{1,1} = \frac{x_{1,0}x_{1,1}}{x_{1,0}} \]

\[ x_{0,0} = g^{x_{1,0}x_{1,1}} \]

**group key is independent of honest user's contribution**
Opening Attack against Contributiveness

- wishes to enforce $x_{0,0} = g^r$ for some $r$ of own choice
- $U_1$ and $U_2$ send their first messages
- asks RevealState($U_1$) and RevealState($U_2$)
- learns $x_{2,0}$, $x_{2,1}$ and computes $x_{1,0}$
- computes $x_{1,1} = r_{x_{1,0}}$

adversary can enforce some chosen value as the group key
Achieving Strong Contributiveness

- opening attacks may reveal all ephemeral secrets
  \( \rightarrow \) contributiveness should not rely on secrets

- **Idea**: use random nonces \( r_1, \ldots, r_n \) and make the value of the derived key depend on each nonce

- not to forget about AKE-security

  - **Attempt 1**: \( K = H(x_{0,0} | r_1 | \ldots | r_n) \)
    AKE-security based on Random Oracles
    the goal is to use standard assumptions

  - **Attempt 2**: \( K = \text{PRF}_{x_{0,0}}(r_1 | \ldots | r_n) \)
    prf collision-resistance wrt. to seeds \textit{and} inputs is undefined
    defined only wrt. to seeds: \( \text{PRF}_{s_1}(v) \neq \text{PRF}_{s_2}(v) \) for any \( s_1 \neq s_2 \)
Iterative PRF execution with embedded nonces

- iterations \( i = 0, \ldots, n \)
- deploys PRF and OWP (one-way premutation)
- public constant input \( v_0 \)
- starts with seed \( x_{0,0} \)
  - iteration 0: \( \rho_0 = \text{PRF}_{x_{0,0}}(v_0) \)
  - iteration \( i \): \( \rho_i = \text{PRF}_{\rho_{i-1} \oplus \text{OWP}(r_i)}(v_0) \)
  - final value: \( K = \rho_n \)

```
input  \( v_0 \)
\downarrow
seed  \( x_{0,0} \) → PRF → \( \oplus \) → PRF → \( \oplus \) → ... → \( \oplus \) → PRF → \( \oplus \) → PRF → K
up    nonces \( r_1 \) → OWP \( r_2 \) → OWP \( r_{n-1} \) → OWP \( r_n \) → OWP
```
Achieving Strong MA and AKE

- based on the compiler by Katz-Shin 2005
- public constants $v_1$ and $v_2$
- Round 2 ends with $U_1$ broadcasts $1|Y|\sigma'_1$

- Round 3
  $U_i$ verifies $\sigma'_1$ and computes $x_{0,0}$ and $K$
  computes $\mu_i = \text{PRF}_K(v_1)$
  computes $\sigma''_i$ on $2|\mu_i|\text{sid}_i|\text{pid}_i$
  broadcasts $2|\sigma''_i$

- Key Computation
  $U_i$ verifies all $\sigma''_j$ using own $\mu_i$
  computes final $K = \text{PRF}_K(v_2)$
  erases state and accepts with $K$
## Efficiency and Security Comparison with *Static* GKE Protocols

<table>
<thead>
<tr>
<th>GKE</th>
<th>Efficiency</th>
<th>Security Goals</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Comm</td>
<td>Comp</td>
<td>AKE</td>
</tr>
<tr>
<td>Abdalla et al.[^{ABC06}]</td>
<td>O(1)</td>
<td>O(n)</td>
<td>w</td>
</tr>
<tr>
<td>Bresson-Catalano[^{BC04}]</td>
<td>O(1)</td>
<td>O(n)</td>
<td>w</td>
</tr>
<tr>
<td>Bresson et al.[^{BCPQ01}]</td>
<td>O(n)</td>
<td>O(n)</td>
<td>w</td>
</tr>
<tr>
<td>Desmedt et al.[^{DPSW06}]</td>
<td>O(1)</td>
<td>O(n)</td>
<td>w</td>
</tr>
<tr>
<td>Dutta et al.[^{DBS04}]</td>
<td>O(1)</td>
<td>O(n)</td>
<td>w</td>
</tr>
<tr>
<td>Katz-Yung[^{KY03}]</td>
<td>O(1)</td>
<td>O(n)</td>
<td>w</td>
</tr>
<tr>
<td>Katz-Shin[^{KS05}] compiler</td>
<td>O(1)</td>
<td>O(n)</td>
<td>s</td>
</tr>
<tr>
<td>TDH1</td>
<td>O(1)</td>
<td>O(n)</td>
<td>s</td>
</tr>
</tbody>
</table>

- w (weak corruptions)
- h (honest participants)
- s (strong corruptions)
- m (malicious participants)
- no opening attacks in \[^{KS05}\]
Summary

• strong corruptions $\rightarrow$ weak corruptions + opening attacks

• stronger security requirements for GKE protocols
  – Strong AKE
  – Strong MA (up to n-2 malicious + n opened)
  – Strong Contributiveness (up to n-1 malicious + n opened)

• Tree Diffie-Hellman Problem $\Leftrightarrow$ DDH for arbitrary full binary trees

• TDH1 protocol
  – achieves all three strong security requirements
  – 3 rounds, standard assumptions

• techniques general an applicable for 2-KE protocols