The Fairy-Ring Dance:
Password Authenticated Key Exchange in a Group

Speaker: Feng Hao

School of Computing Science
Newcastle University, UK

ASIACCS IoTPTS’15, Singapore
Joint work with:

- **Xun Yi** (RMIT University, Australia, UK)
- **Liqun Chen** (HP Labs, UK)
- **Siamak F. Shahandashti** (Newcastle University, UK)
Internet of Things

All communications via (insecure) Internet
Secure Group Communication

- The need for secure group communication
  - One group key is easier to manage than many pairwise keys
- Where to bootstrap the trust?
  - Not from PKI, as we want to avoid it
  - Instead, from a common (low-entropy) password
- In practice, one enters a short code to each device
  - No pre-installed certificates or secrets required
Password Authenticated Key Exchange

- Extensively studied since 1992
- Several solutions available: EKE, SPEKE, SRP-6, J-PAKE
Group PAKE

- A natural extension from two-party to multi-party
- However, not a trivial extension
  - Group PAKE is more difficult to design than two-party PAKE
  - Very few studies on Group PAKE so far
- However, IoT may prove a killer app for Group PAKE
Challenge in designing Group PAKE

- Security requirements have been well understood
  - Similar to two-party PAKE
- One practical challenge is to make it round-efficient
  - Computation improves rapidly over time (Moore’s law)
  - Communication improves only modestly
  - The rounds always stay the same
- The overall latency is mainly determined by the slowest responder
Round efficiency

- Many Group PAKE protocols require $O(n)$ rounds
- Best round efficiency so far: constant 4 rounds (Abdalla et al, PKC’06)
- Here, we show how to achieve 2 rounds (theoretical best)
The topology of group communications

- Previous designs generally assume a circle
  - A participant only talks to two neighbors (left and right)
  - Essentially, following the same topology as Burmester-Desmedt (Eurocrypt'95)
- But we will use a different topology: fully-connected graph
  - No increase in the communication complexity
  - All data is broadcasted in the public
Fairy-Ring Dance

- A traditional Scottish dance
- Men and women form a circle, and dance in rotation
- Everyone dances with everyone else

(Source: YouTube)
A more technical view

- Run two processes in parallel
  - Pairwise PAKE sessions (inner dash lines)
  - One group session establishment (outer circle)
Two concrete instantiations

- **SPEKE+**
  - Use SPEKE for pairwise PAKE sessions
  - Use (modified) Burmester-Desmedt for group session

- **J-PAKE+**
  - Use J-PAKE for pairwise PAKE sessions
  - Use (modified) Burmester-Desmedt for group session
First Group PAKE scheme: SPEKE+

- Combining SPEKE and BD with optimal round-efficiency

**SPEKE**
- Proposed by Jablon in 1996
- Used in commercial applications (Blackberry)

**BD**
- Proposed by Burmester and Desmedt in 1995
- Almost universally used in group key exchange schemes
- But it’s unauthenticated
**SPEKE protocol [Jablon’96]**

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $x \in R \ Z_q$</td>
<td>$X = g^x$</td>
</tr>
<tr>
<td>2. Verify $Y \in [2, p - 2]$</td>
<td>$Y = g^y$</td>
</tr>
<tr>
<td>$k = H(Y^x) = H(g^{xy})$</td>
<td>$k = H(X^y) = H(g^{xy})$</td>
</tr>
</tbody>
</table>

Explicit key confirmation (optional)

- Use a safe prime $p = 2q + 1$
- Use a password-derived generator: $g = s^2$ (later changed to $g = H(s)^2$)
BD protocol [Burmester-Desmedt’95]

**Round 1**

*Every participant* $P_i$ *selects* $y_i \in_R [0, q−1]$ *and broadcasts* $g^{y_i}$.

Everyone can compute $g^{z_i} = g^{y_{i+1}} / g^{y_{i−1}}$.

**Round 2**

*Every participant* $P_i$ *broadcasts* $(g^{z_i})^{y_i}$.

**Group session key:**

$$K_i = (g^{y_{i−1}})^{n \cdot y_i} \cdot (g^{z_i} y_i)^{n−1} \cdot (g^{z_{i+1} y_{i+1}})^{n−2} \ldots (g^{z_{i−2} y_{i−2}})$$

$$= g^{y_1 \cdot y_2 + y_2 \cdot y_3 + \cdots + y_n \cdot y_1}$$
**SPEKE+ (Two rounds with key confirmation)**

**Round 1**

*Every participant* $P_i$ *selects* $x_i \in R [1, q - 1]$, $y_i \in R [0, q - 1]$ *and broadcasts* $g_s^{x_i}$, $g^{y_i}$ *together with ZKP for* $y_i$.

Everyone can compute $g^{z_i} = g^{y_{i+1}} / g^{y_{i-1}}$.

**Round 2**

*Every participant* $P_i$ *broadcasts* $(g^{z_i})^{y_i}$ *and a ZKP for proving* $\log_{g^{z_i}}(g^{z_i})^{y_i} = \log_g g^{z_i}$. *Furthermore,* $P_i$ *computes two pairwise keys with each of the rest participants: 1)*

$\kappa_{ij}^{\text{MAC}} = H(g_s^{x_i \cdot x_j} \parallel \text{"MAC"}); 2) \ \kappa_{ij}^{\text{KC}} = H(g_s^{x_i \cdot x_j} \parallel \text{"KC"})$. *The 1st key is used to authenticate the group key while the 2nd key is used for key confirmation in pairwise PAKE sessions. (more details in paper)*

$$K_i = (g^{y_{i-1}})^n \cdot y_i \cdot (g^{z_i \cdot y_i})^{n-1} \cdot (g^{z_{i+1} \cdot y_{i+1}})^{n-2} \ldots (g^{z_{i-2} \cdot y_{i-2}})$$

$$= g^{y_1 \cdot y_2 + y_2 \cdot y_3 + \ldots + y_n \cdot y_1}$$
Second Group PAKE scheme: J-PAKE+

- J-PAKE [Hao, Ryan, 2008]
  - Included in open source libraries (OpenSSL, Bouncycastle, NSS)
  - Used in commercial applications (Firefox, Palemoon, Nest)
  - Accepted by ISO/IEC 11770-4 standard (in process)

- J-PAKE+ (our new contribution)
  - A group variant of J-PAKE
J-PAKE+

- Original two-party J-PAKE
  - Two rounds with implicit key confirmation
  - Three rounds with explicit key confirmation

- Multi-party J-PAKE+
  - Combining J-PAKE and BD with optimal round-efficiency
  - Three rounds with explicit key confirmation
  - Based on the same Fairy-Ring Dance construction
  - Protocol details omitted in this talk (see paper)
Implementation of SPEKE+ and J-PAKE+

- Implemented both protocols in pure Java
- Used only the standard BigInteger class for all the modular exponentiations
- Chose the 2048-bit group setting
- Source code available at: https://github.com/FairyRing/SourceCode
Comparing latency between SPEKE+ and J-PAKE+

Tested on 2.93 GHz PC with 4 GB RAM running 64-bit Windows 7
Breakdown of costs in SPEKE+

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>Complexity</th>
<th>No of exponentiations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Computation in R1</td>
<td>$O(1)$</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Verification after R1</td>
<td>$O(n)$</td>
<td>$(n-1) \times 2.215$</td>
</tr>
<tr>
<td>3</td>
<td>Computation in R2</td>
<td>$O(n)$</td>
<td>$3 + (n-1) \times 1$</td>
</tr>
<tr>
<td>4</td>
<td>Verification after R2</td>
<td>$O(n)$</td>
<td>$(n-1) \times 3.25$</td>
</tr>
<tr>
<td>5</td>
<td>Compute group key</td>
<td>$O(1)$</td>
<td>1</td>
</tr>
</tbody>
</table>

Cost breakdown in SPEKE+

- Cost of computation in round 1
- Cost of verification after round 1
- Cost of computation in round 2
- Cost of verification after round 2
- Cost of computation of group key

Latency (ms) vs. Number of the participants in the group
### Breakdown of costs in J-PAKE+

<table>
<thead>
<tr>
<th></th>
<th>Cost breakdown</th>
<th>Complexity</th>
<th>No of exponentiations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Computation in R1</td>
<td>$O(n)$</td>
<td>$2 + (n-1) \times 4$</td>
</tr>
<tr>
<td>2</td>
<td>Verification after R1</td>
<td>$O(n)$</td>
<td>$(n-1) \times 9$</td>
</tr>
<tr>
<td>3</td>
<td>Computation in R2</td>
<td>$O(n)$</td>
<td>$(n-1) \times 2$</td>
</tr>
<tr>
<td>4</td>
<td>Verification after R2</td>
<td>$O(n)$</td>
<td>$(n-1) \times 4$</td>
</tr>
<tr>
<td>5</td>
<td>Computation in R3</td>
<td>$O(n)$</td>
<td>$5 + (n-1) \times 2$</td>
</tr>
<tr>
<td>6</td>
<td>Verification after R3</td>
<td>$O(n)$</td>
<td>$(n-1) \times 5$</td>
</tr>
<tr>
<td>7</td>
<td>Compute group key</td>
<td>$O(1)$</td>
<td>1</td>
</tr>
</tbody>
</table>

Cost breakdown in J–PAKE+

![Cost breakdown in J–PAKE+](image_url)
Security properties of SPEKE+ and J-PAKE+

- Off-line dictionary attack resistance
  - Reducible to the underlying PAKE
- Known-session security
  - Reducible to the underlying PAKE
- Forward secrecy
  - Reducible to the underlying PAKE
- On-line dictionary attack resistance
  - Reducible to the underlying PAKE
  - However, the number of guesses increases to $\alpha \times (n - \alpha)$ where $\alpha$ is the number of legitimate participants and $n$ is the total number of participants
The Good and Bad about Fairy-Ring Dance

- **The Good**
  - Preserves the round efficiency in the optimal way
  - Allows us to achieve better round efficiency than previous works
  - Pushes the known best result to 2 rounds (theoretical best)

- **The Bad**
  - More than one password guesses in on-line attack: ideally, should be exactly one
  - $O(n)$ computation per participant: ideally, should be $O(1)$

- However, need to put the “Bad” into a practical perspective
  - Not any serious concern for a small-medium sized group
  - Overall, a worthwhile trade-off
Research on two-party PAKE started from 1990s
  - Extensively studied in the past 20 years
  - Practical deployment of PAKE only takes off in recent years

Research multi-party Group PAKE has been lagging far behind
  - Very few studies in this area
  - No Group PAKE has been used in any practical applications
  - However, the IoT may change the landscape

We contribute two Group PAKEs
  - Both are sufficiently efficient for practical use
  - Open source implementations available
Thank you!

For more technical details, see