Revisiting pairing based group key exchange

Yvo Desmedt

BT Chair of Information Security

University College London

Tanja Lange

TU Eindhoven

28.01.2008

Group key exchange (GKE)

- Situations in which a (temporarily fixed) group of people should communicate quickly in a secured manner:
  - secured phone or video-conferencing,
  - encryption of message for a group of users, e.g. for task oriented groups.
- Efficiency reasons dictate use of symmetric crypto for encryption.
- Each message should be encrypted only once, so all users should have the same key.

Group key exchange protocols manage that each user gets same key.

Obvious approach

- selected user picks random key
- does DH key exchange with every user
- sends selected key over this channel.

Advantages:

- Easy to set up.
- Most users need to compute 2 exponentiations, 1 key derivation, 1 symmetric decryption.

Disadvantages:

- One user needs to compute n exponentiations, $n - 1$ key derivations, $n - 1$ symmetric encryptions, where n is the size of the group.
- Very unbalanced.

Burmester-Desmedt I GKE

Constant (2) rounds for any number of users.
Users $U_1, \ldots, U_n$, subgroup $\langle g \rangle \subset (\mathbb{Z}/p\mathbb{Z})^*$ of order ℓ.

Step 1 Each U_i selects $r_i \in_R [1..\ell]$; computes and sends to his direct neighbors $Z_i = g^{r_i}$.

Step 2 Each U_i computes and broadcasts

$$X_i = \left(\frac{Z_{i+1}}{Z_{i-1}}\right)^{r_i},$$

where the indices are taken modulo n.

Step 3 Each U_i computes the conference key,

$$K_i = Z_{i-1}^{nr_i} X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i-2},$$

where the indices are taken modulo n.

Burmester-Desmedt I GKE

- All users compute the same key
$$K = K_i = g^{r_1 r_2 + r_2 r_3 + \cdots r_n r_1}.$$
- Note on efficiency: The key is computed in a Horner like scheme as
$D \leftarrow Z_{i-1}^{r_i}, K \leftarrow D$
for $j = 0$ to $n-2$
$$D \leftarrow D \cdot X_{i+j} \qquad //D = Z_{i-1}^{r_i} X_i X_{i+1} \cdots X_{i+j}$$
$$K \leftarrow K \cdot D \qquad //K = Z_{i-1}^{(j+2)r_i} X_i^{j+1} X_{i+1}^{j} \cdots X_{i+j}$$
i.e., the whole computation takes 3 exponentiations, 1 inversion, and $2n-1$ multiplications.
- Published by Burmester and Desmedt in Eurocrypt'94 (preproceedings version had security proof, omitted in full version), see Information Processing Letters **94**, 2005.

Authenticated GKE

- Katz and Yung (Crypto'03) proposed a compiler to turn GKE into authenticated GKE, they apply it to BD1 as example.
- Essential ingredients:
  - include group identifier,
  - include message number,
  - sign each message.
- Protocols using the Katz-Yung compiler cannot run faster than $\Theta(n)$.
- Desmedt, Lange, and Burmester (FC'07) generalized compiler to other architectures, e.g. to BD2 which has a tree architecture and thus runs in $O(\log n)$.
- Interesting feature of DLB: only signatures on used messages are checked.

Pairing-based crypto

- Joux (ANTS 2000) introduced 1 round tripartite key exchange, based on new primitive bilinear map/pairing.

Pairing-based crypto

- Joux (ANTS 2000) introduced 1 round tripartite key exchange, based on new primitive bilinear map/pairing.
- Pairings are an interesting and still growing area of crypto, see flyers for Pairings'08 outside!

Pairing-based crypto

- Joux (ANTS 2000) introduced 1 round tripartite key exchange, based on new primitive bilinear map/pairing.
- Essential property of pairings $\hat{e}: G_1 \times G_1 \to G_2$

$$\hat{e}(aP, bQ) = (\hat{e}(P, Q))^{ab}$$

  for any integers a, b.
- Key exchange (from Alice's perspective):
  - Send $A = aP$ to Bob and Charlie.
  - Receive $B = bP$ and $C = cP$.
  - Compute $(\hat{e}(B, C))^a = (\hat{e}(P, P))^{abc}$
- Can be made authenticated etc.

Pairings

- Common instantiation:
  - G_1 subgroup of points on elliptic curve E over finite field $\mathbb{F}_q$ (therefore $(G_1, +)$).
  - $\hat{e}$ modified Tate (or Weil) pairing.
  - G_2 subgroup of multiplicative group over finite field $\mathbb{F}_{q^k}$ (therefore $(G_2, \cdot)$).
- …or at least close to this …
- Performance depends a lot on choice of curve and field.
- Early results (e.g. using the parameters from Boneh/Franklin ID-based encryption) have 1 pairing > 10 scalar multiplications.
- Better parameters (e.g. Granger/Page/Stam'06) 1 pairing $\sim$ 3-4 scalar multiplications.

BD1 using pairings

Choi, Hwang, and Lee PKC 2004.
Users $U_1, \ldots, U_n$, groups $G_1 = \langle P \rangle, G_2$ of order ℓ, pairing $\hat{e}$.

Step 1 Each U_i selects $r_i \in_R [1..\ell]$; computes and sends to his direct neighbors $Z_i = r_i P$.

Step 2 Each user U_i computes and broadcasts

$$X_i = \hat{e}\left(r_i(Z_{i+2} - Z_{i-1}), Z_{i+1}\right),$$

where the indices are taken modulo n.

Step 3 Each U_i computes the conference key,

$$K_i = \hat{e}(r_i Z_{i-1}, Z_{i+1})^n X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i-2},$$

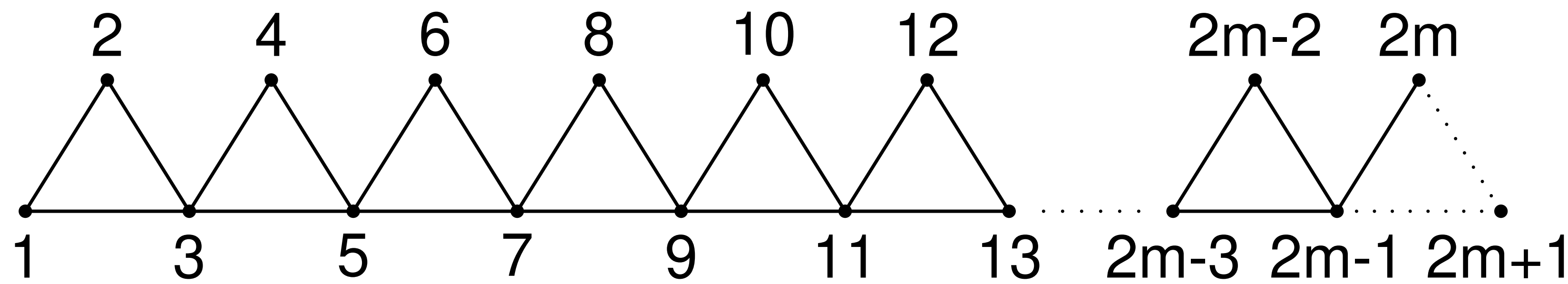
where the indices are taken modulo n.

Choi, Hwang, and Lee '04

- Joint key is
$$K = \hat{e}(P,P)^{r_1 r_2 r_3 + r_2 r_3 r_4 + \cdots + r_{n-1} r_n r_1 + r_n r_1 r_2}.$$
- Security is based on bilinear DDHP.
- Each user computes 2 pairings, 3 scalar multiplications in G_1, 1 add in G_1, 2(n-1) mult. in G_2; obvious improvement to 2 scalar multiplications.
- Faster than classical BD1 if this is cheaper than 3 exponentiations, 1 inversion, and 2n-1 multiplications. (Note that one can use G_1 instead of $(\mathbb{Z}/p\mathbb{Z})^*$.)
- In authenticated version (using DLB compiler) each user
  - computes 2 signatures,
  - checks n+1 signatures.

Our pairing based scheme I

- Arrangement of users as in picture:



- The even indexed users have easier load, application is most beneficial if some users have less computational resources; in particular they don't need broadcast. Typical scenarios are
  - Odd indexes are servers that are permanently online, even indexes are home users with less bandwidth (in particular in the upstream).
  - Phone conference with some users on mobile without access to more computational power.

Our pairing-based scheme II

Users $U_1, \ldots, U_n$, groups $G_1 = \langle P \rangle, G_2$ of order ℓ, pairing $\hat{e}$, $n = 2m + a, a \in \{0, 1\}$.

Step 1 Each selects $r_i \in_R [1..\ell]$, computes and sends to his direct neighbors $Z_i = r_i P$.

Step 2 Each U_i, $i = 3, \ldots, 2m-1$, odd, computes and broadcasts

$$X_i = \left(\frac{\hat{e}(Z_{i+1}, Z_{i+2})}{\hat{e}(Z_{i-2}, Z_{i-1})} \right)^{r_i},$$

where the indices are taken modulo n.

ctd.

Step 3 Each U_i, $i = 1, \ldots, n$, computes the conference key,

$$K_i = \begin{cases} (\hat{e}(Z_2, Z_3))^{(m-1)r_1} X_3^{m-2} X_5^{m-3} \cdots X_{2m-3}, & \text{when } i = 1, \\ \\ T_i^{(m-1)r_i} (X_3 X_5^2 \cdots X_{2j-1}^{j-1})^{-1} (X_{2j+1}^{m-j-1} X_{2j+3}^{m-j-2} \cdots X_{2m-3}), & \text{else,} \end{cases}$$

where $i = 2j + k, k \in \{0, 1\}$ and $T_i = \hat{e}(Z_{i-1}, Z_{i+1})$ for i even and $T_i = \hat{e}(Z_{i-2}, Z_{i-1})$ for odd i.

All users compute the same key, $K = \hat{e}(P, P)^d$ for

$$d = r_1 r_2 r_3 + r_3 r_4 r_5 + r_5 r_6 r_7 + \cdots + r_{2m-3} r_{2m-2} r_{2m-1}.$$

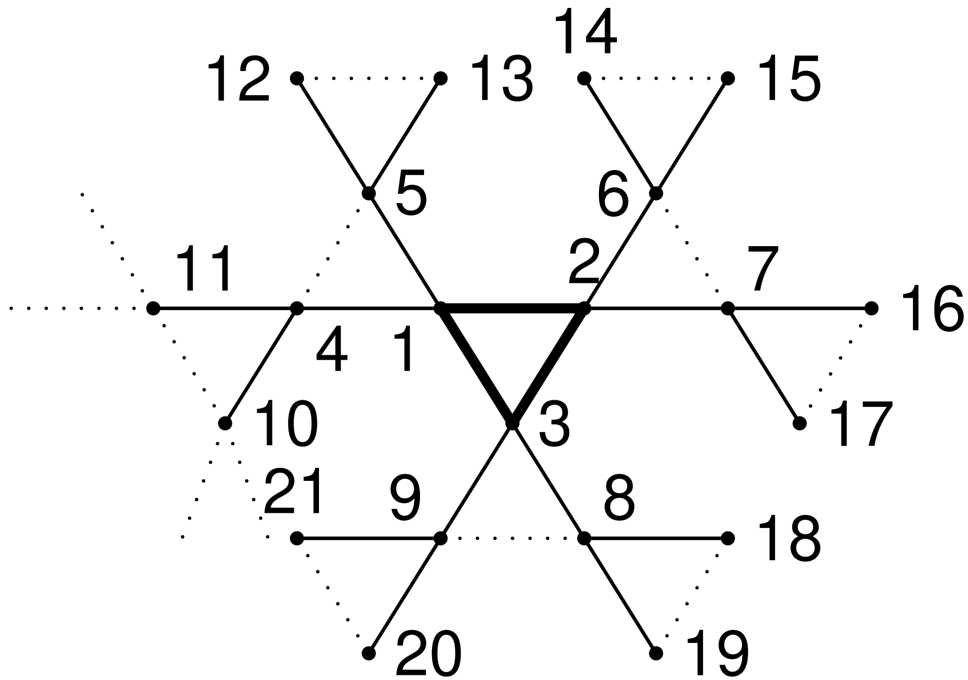
Example

$$
\begin{aligned}
K_1 &= \hat{e}(Z_2, Z_3)^{2r_1} X_3 = \hat{e}(P, P)^{2r_1r_2r_3 - r_1r_2r_3 + r_3r_4r_5} = \hat{e}(P, P)^{r_1r_2r_3 + r_3r_4r_5} \\
K_2 &= \hat{e}(Z_1, Z_3)^{2r_2} X_3 = \hat{e}(P, P)^{2r_1r_2r_3 - r_1r_2r_3 + r_3r_4r_5} = \hat{e}(P, P)^{r_1r_2r_3 + r_3r_4r_5} \\
K_3 &= \hat{e}(Z_1, Z_2)^{2r_3} X_3 = \hat{e}(P, P)^{2r_1r_2r_3 - r_1r_2r_3 + r_3r_4r_5} = \hat{e}(P, P)^{r_1r_2r_3 + r_3r_4r_5} \\
K_4 &= \hat{e}(Z_3, Z_5)^{2r_4} X_3^{-1} = \hat{e}(P, P)^{2r_3r_4r_5 - (-r_1r_2r_3 + r_3r_4r_5)} = \hat{e}(P, P)^{r_1r_2r_3 + r_3r_4r_5} \\
K_5 &= \hat{e}(Z_3, Z_4)^{2r_5} X_3^{-1} = \hat{e}(P, P)^{2r_3r_4r_5 - (-r_1r_2r_3 + r_3r_4r_5)} = \hat{e}(P, P)^{r_1r_2r_3 + r_3r_4r_5} \\
K_6 &= \hat{e}(Z_1, Z_5)^{2r_6} (X_3 X_5^2)^{-1} = \hat{e}(P, P)^{2r_1r_5r_6 - (-r_1r_2r_3 + r_3r_4r_5 + 2(-r_3r_4r_5 + r_5r_6r_1))} \\
&= \hat{e}(P, P)^{r_1r_2r_3 + r_3r_4r_5}
\end{aligned}
$$

Analysis

- Security is based on bilinear DDHP.
- Each user computes 1 pairing, 2 scalar multiplications in G_1, 2(m-1) mult. in G_2; odd indices compute 1 pairing, 1 exponentiation in G_1, and 1 inversion extra.
- No slower than previous proposal; much faster for even indices.
- In authenticated version (using DLB compiler) each odd user
  - computes 2 signatures,
  - checks n/2+2 signatures.
- In authenticated version (using DLB compiler) each even user
  - computes 1 signature,
  - checks n/2-2 signatures.

Combination of BD2 and pairings



- Works, is faster, looks nice ;–)
- A bit too messy to be explained in a talk.
- Will be included in full version of the paper.

The END

Thanks!