Flexible Group Key Exchange with On-Demand Computation of Subgroup Keys

Michel Abdalla[1], Celine Chevalier[2], Mark Manulis[3], David Pointcheval[1]

[1]École Normale Supérieure CNRS-INRIA, Paris, France

[2]Telecom ParisTech, France

[3]Cryptographic Protocols Group, TU Darmstadt & CASED, Germany

Group Key Exchange

Users in $\mathbf{U} = \{U_1, \ldots, U_N\}$ run a **Group Key Exchange (GKE)** protocol and compute a session group key k *indistinguishable from* $k^* \in_R \{0,1\}^\kappa$

U_1

U_i

U_N

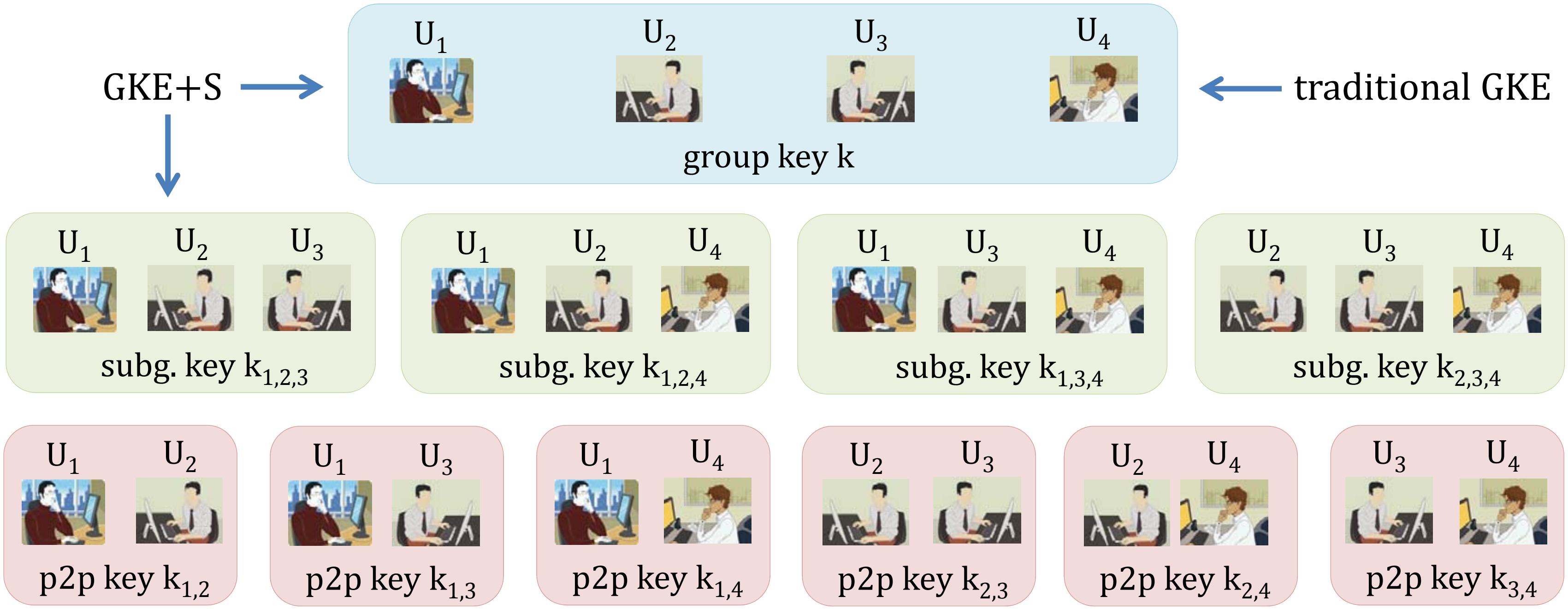
accept k_1 accept k_i accept k_N

main building block for secure group communication

Correctness requires that $k_1 = k_2 = \ldots = k_N$

Flexible Group Key Exchange

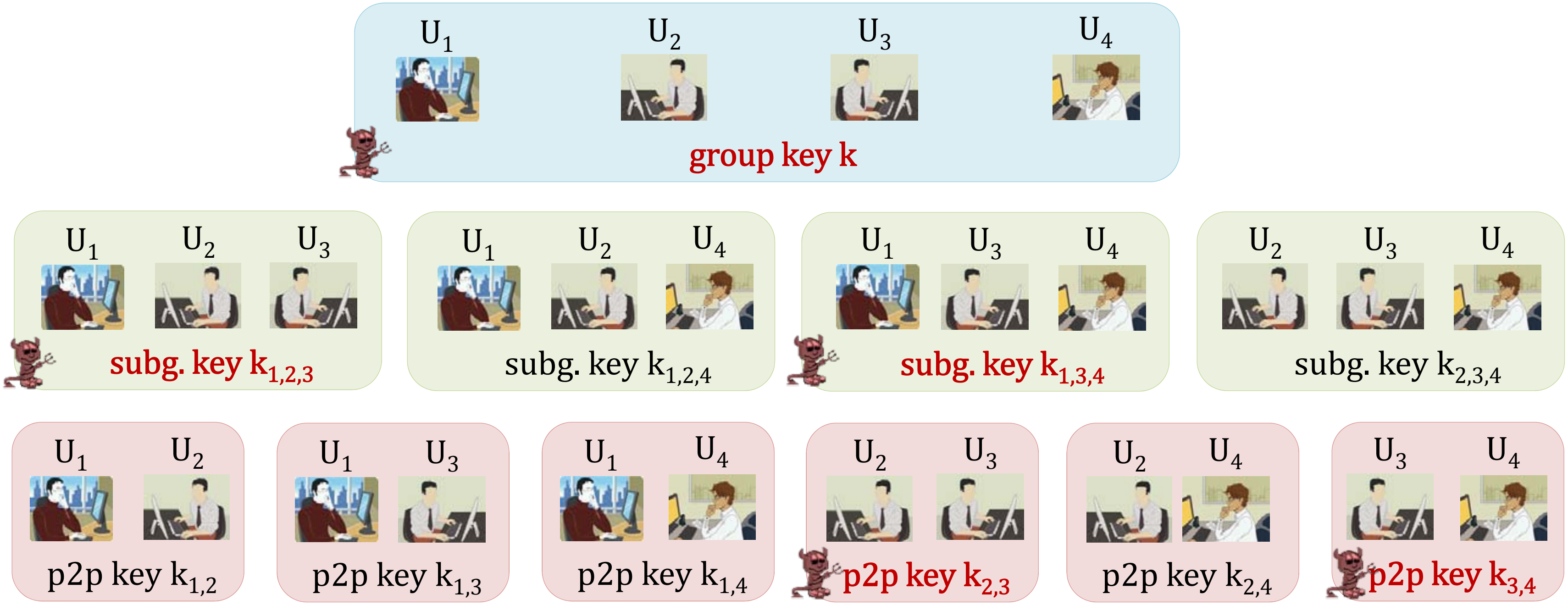
Goal Extend the notion of GKE towards computation of subgroup/p2p keys.



Naïve solution Each subgroup executes its own GKE/2KE session on-demand.

Is it possible to compute subgroup/p2p keys in some optimized, more efficient way?

Challenge 1: Independence of Subgroup Keys

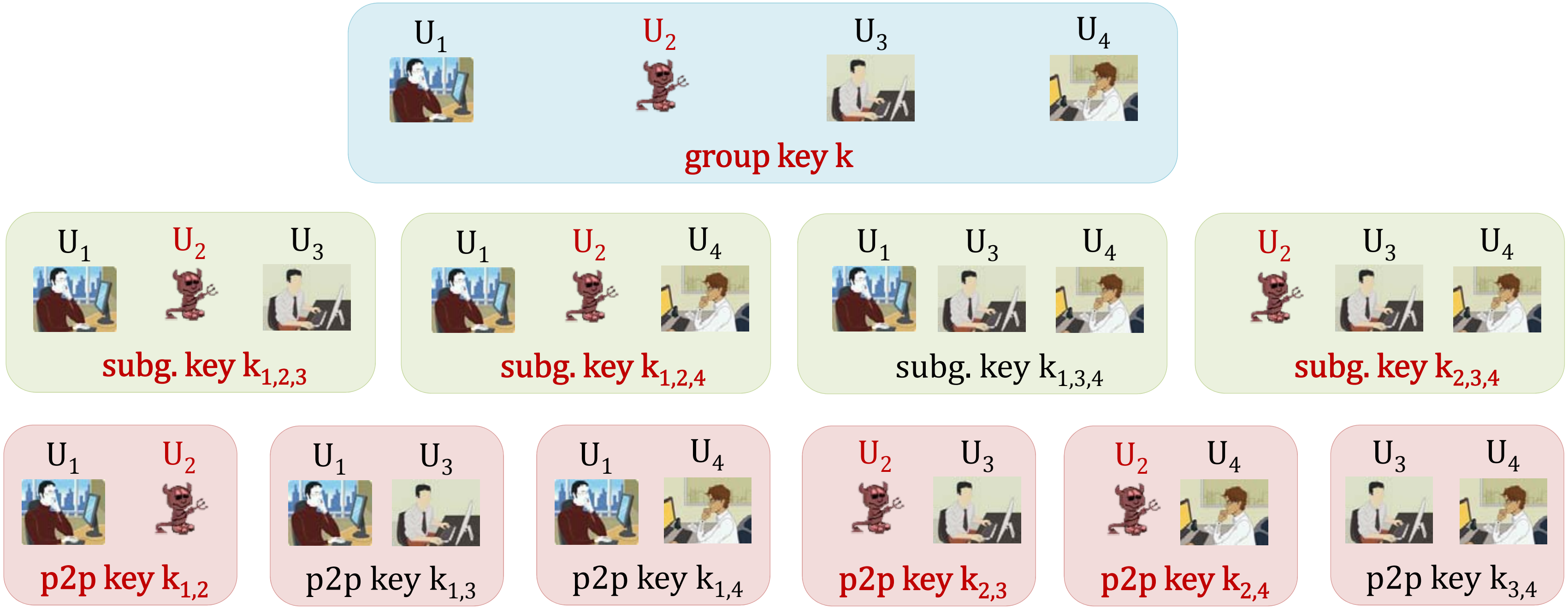


Adversary $\mathcal{A}$ may learn some session keys (incl. the group key).

Still, security of other unknown subgroup/p2p keys should be preserved.

Session keys must be independent (indistinguishable from random keys).

Challenge 2: Insider/Collusion Attacks



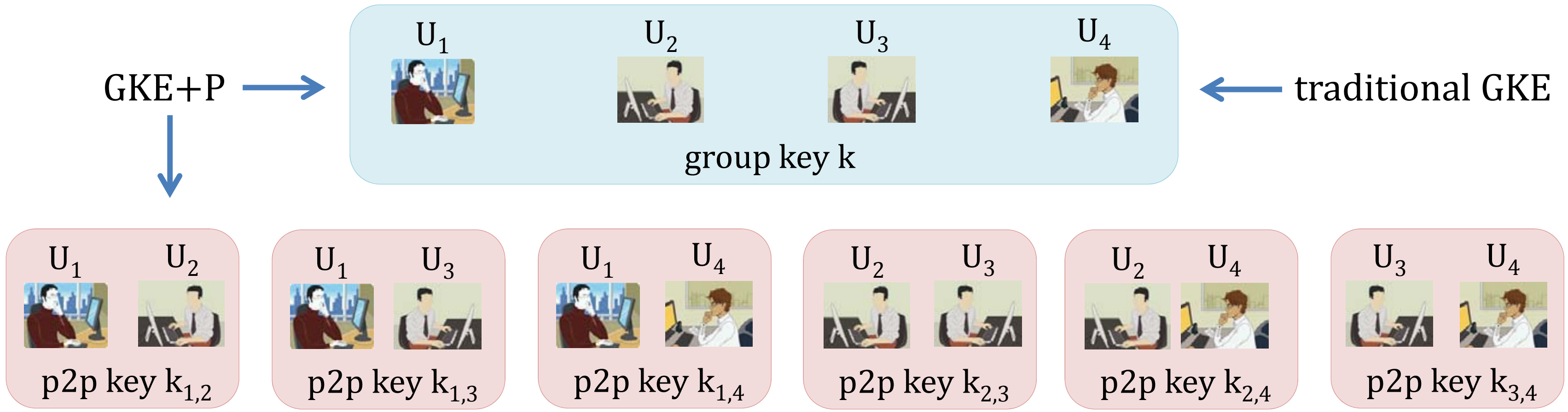
Adversary $\mathcal{A}$ may be a group member and misbehave during the protocol execution. Still, security of subgroup keys (where $\mathcal{A}$ is not a member) should be preserved.

Independence of (sub)group keys must hold in case of insider /collusion attacks.

GKE+P Protocols

GKE+P GKE with On-Demand Derivation of P2P Keys [Man09]

Can be seen as a *special case* of GKE+S.



Many GKE protocols extend the classical Diffie-Hellman method to a group setting.

The **group key k** is derived from some element **$k' = f(g, x_1, \ldots, x_N)$**

for some function $f : \mathbb{G} \times \mathbb{Z}_Q^N \longrightarrow \mathbb{G}$, where $x_i \in \mathbb{Z}_Q$ is an exponent chosen by U_i.

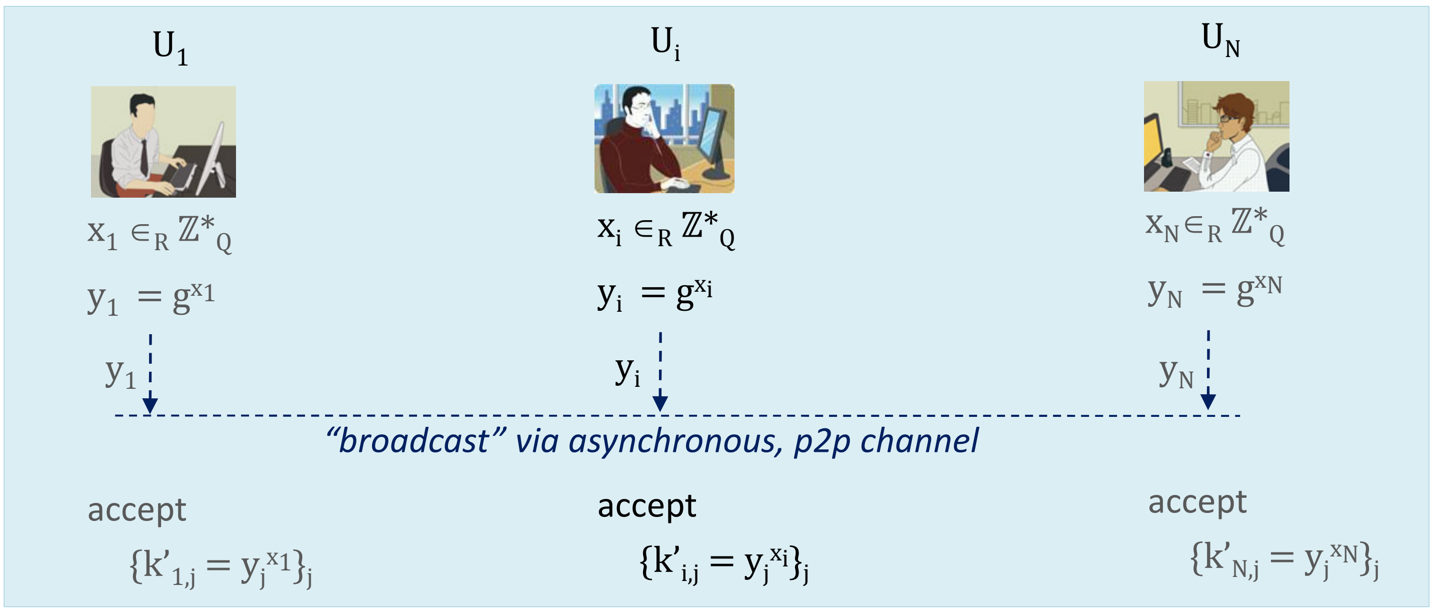
Is it possible to *re-use* exponents x_i and x_j to derive p2p keys from $g^{x_i x_j}$?

Parallel Diffie-Hellman Key Exchange

As a basic tool to derive p2p keys we want to use the *parallel* version of DHKE.

Parallel DHKE (PDHKE)

Let $\mathbf{U} = \{U_1, \ldots, U_N\}$ be a set of users (their *unique* identities).

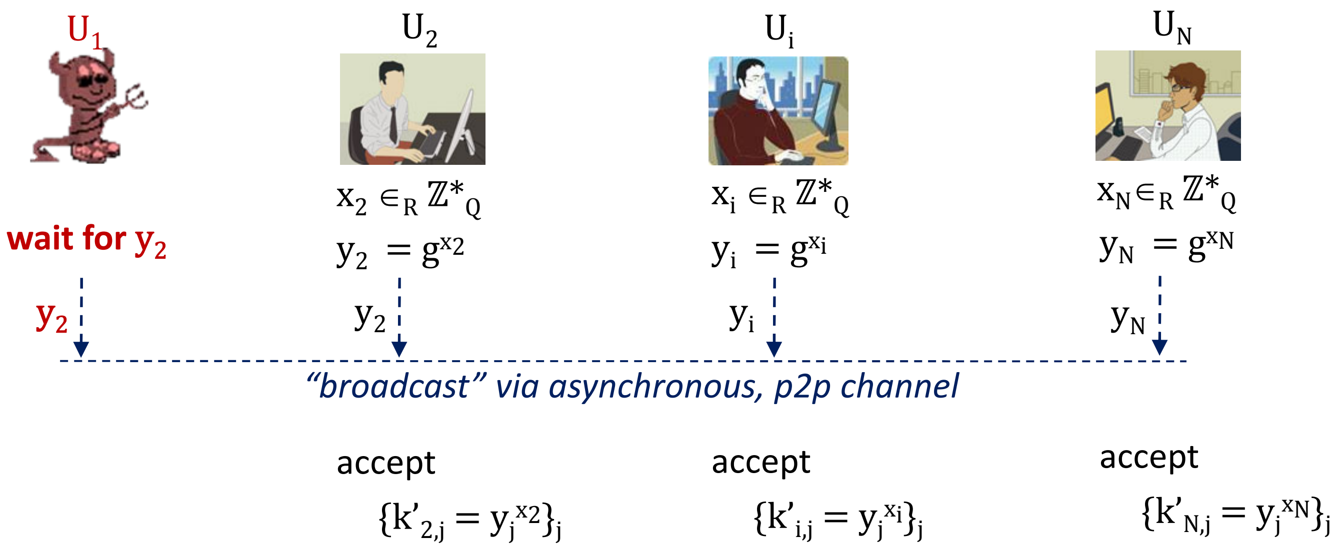


Allows U_i to compute $k'_{i,1} = g^{x_i x_1}, k'_{i,2} = g^{x_i x_2}, \ldots, k'_{i,N} = g^{x_i x_N}$. However,...

Simple Insider Attack on PDHKE

Recall that P2P keys should remain independent.

Insider Attack on PDHKE



Although $\mathcal{A}$ does not learn x_2 we have $k'_{i,1} = k'_{i,2} = g^{x_i x_2}$ for all U_i.

Exposure of any $k'_{i,1}$ to $\mathcal{A}$ reveals $k'_{i,2}$, which however should remain secret.

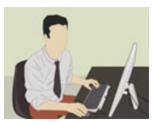
Hash-based Key Derivation for PDHKE

The problem can be fixed by appropriate key derivation function applied to $k'_{i,j}$.

Hash-based Key Derivation for PDHKE

Let $H_p : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$ be a cryptographic hash function (random oracle).

U_i



$x_i \in_R \mathbb{Z}^*_Q$

$y_i = g^{x_i}$ — y_i ⟶

⟵ $\{y_j\}_j$

$\{k'_{i,j} = y_j^{x_i}\}_j$

accept

$\{k_{i,j} = H_p(k'_{i,j}, (U_i, y_i), (U_j, y_j))\}_j$

$k_{i,j} = H_p(k'_{i,j}, (U_i, y_i), (U_j, y_j))$ 

for any U_i, U_j the input order to H is determined by $i < j$ (s.t. $k_{i,j} = k_{j,i}$)

uniqueness of U_i, U_j

⇒ uniqueness of hash inputs $H_p(*, (U_i, *), (U_j, *))$

uniqueness of y_i per session

⇒ independence of p2p session keys $\{k_{i,j}\}_j$ of U_i

(in the random oracle model)

This allows us to derive **independent p2p keys for any pair** (U_i, U_j).

Can we integrate PDHKE into a GKE protocol?

Integration into Burmester-Desmedt GKE Fails

Burmester-Desmedt (BD) GKE[BD94] Cyclic DL-hard group $\mathbb{G} = (g, P, Q)$.

Users $U_1, \ldots, U_N$ are arranged into a *cycle* such that $U_0 = U_N$, $U_{N+1} = U_1$.

U_{i-1}	U_i	U_{i+1}
		
$x_{i-1} \in_R \mathbb{Z}_Q$	$x_i \in_R \mathbb{Z}_Q$	$x_{i+1} \in_R \mathbb{Z}_Q$
$y_{i-1} = g^{x_{i-1}}$	$y_i = g^{x_i}$	$y_{i+1} = g^{x_{i+1}}$

Round 1

$z_{i-1} = (y_i/y_{i-2})^{x_{i-1}}$	$z_i = (y_{i+1}/y_{i-1})^{x_i}$	$z_{i+1} = (y_{i+2}/y_i)^{x_{i+1}}$

Round 2

$$k'_i := y_{i-1}^{Nx_i} z_i^{N-1} z_{i+1}^{N-2} \ldots z_{i+N-2} = g^{x_1x_2 + x_2x_3 + \ldots + x_{N-1}x_N}$$

group key $k_i := H_g(g^{x_1x_2 + x_2x_3 + \ldots + x_{N-1}x_N}, (U_1, y_1), \ldots, (U_N, y_N))$

p2p keys $k_{i,j} := H_p(g^{x_ix_j}, (U_i, y_i), (U_j, y_j))$

However,...

Problem and Solution

P2P keys are not independent[Ma09] Each U_i sends $\mathbf{z_i = (y_{i+1}/y_{i-1})^{x_i} = g^{x_i x_{i+1}}/g^{x_{i-1} x_i}}$.
U_{i-1} can compute $g^{x_i x_{i+1}}$ and thus derive the p2p key $k_{i,i+1}$ shared between U_i and U_{i+1}.

Our Solution – modified BD (mBD) Use hash function $H : \mathbb{G} \longrightarrow \{0,1\}^\kappa$.

Let $sid_i = ((U_1, y_1), \ldots, (U_N, y_N))$ known to each U_i after first BD round.
In the second round U_i computes

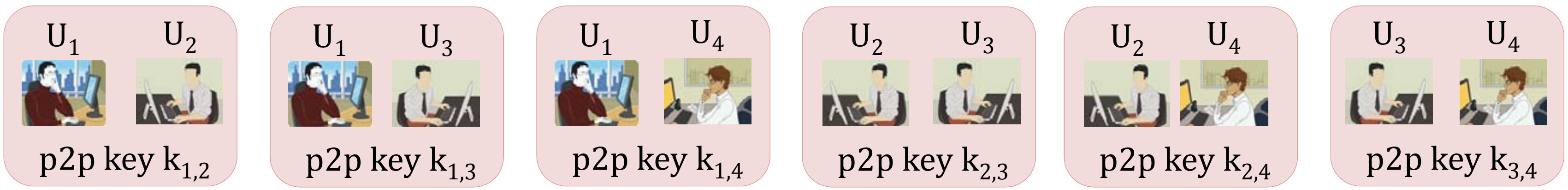
$$\mathbf{z_{i-1,i} = H(y_{i-1}^{x_i}, sid_i)}\ ,\ \mathbf{z_{i,i+1} = H(y_{i+1}^{x_i}, sid_i)} \text{ and broadcasts } \mathbf{z_i = z_{i,i-1} \oplus z_{i,i+1}}.$$

From $z_{i-1,i}$ and $z_1, \ldots, z_N$ each U_i can recover $z_{1,2}, z_{2,3}, \ldots, z_{N,1}$ (via iterated $\oplus$).
In **mBD+P** users derive: group key $k_i = H_g(\mathbf{z_{1,2}}, \ldots, \mathbf{z_{N,1}}, (U_1, y_1), \ldots, (U_N, y_N))$
p2p keys $k_{i,j} = H_p(g^{x_i x_j}, (U_i, y_i), (U_j, y_j))$

Knowledge of $z_{1,2}, \ldots, z_{N,1}$ is *not* sufficient for the computation of any $g^{x_i x_j}$.
In the paper we prove security of mBD+P using **Gap Diffie-Hellman** assumption.

Extension to GKE+S

GKE+P allows any pair of users to derive their p2p key without any interaction.



Can we extend GKE+P towards derivation of subgroup keys?

(Bad News) **We cannot not derive subgroup keys in a *non-interactive* way.**

Due to long-standing open problem **One-Round GKE with Forward Secrecy**

(Good News) **We can compute subgroup keys with *minimum* communication effort.**

Our mBD+S protocol takes only **one (additional) round per subgroup.**

(Unauthenticated) mBD+S

GKE Stage 2 rounds as in mBD+P. Users in **U** compute the group key.

U_{i-1} 

$x_{i-1} \in_R \mathbb{Z}_Q;\ y_{i-1} = g^{x_{i-1}}$

$z_{i-1} = H(y_i^{x_{i-1}}, sid_{i-1}) \oplus H(y_{i-2}^{x_{i-1}}, sid_{i-1})$

U_i 

$x_i \in_R \mathbb{Z}_Q;\ y_i = g^{x_i}$

$z_i = H(y_{i+1}^{x_i}, sid_i) \oplus H(y_{i-1}^{x_i}, sid_i)$

U_{i+1} 

$x_{i+1} \in_R \mathbb{Z}_Q;\ y_{i+1} = g^{x_{i+1}}$

$z_{i+1} = H(y_{i+2}^{x_{i+1}}, sid_{i+1}) \oplus H(y_i^{x_{i+1}}, sid_{i+1})$

$$z_1 \oplus \cdots \oplus z_N \stackrel{?}{=} 0\ ;\ \text{recover } z_{1,2}, \ldots, z_{N,1}$$

$$\text{group key } k_i := H(z_{1,2}, \ldots, z_{N,1}, (U_1, y_1), \ldots, (U_N, y_N))$$

Subgroup Stage 1 round. Users in $\mathbf{S} \subset \mathbf{U}$, $|\mathbf{S}| = M$, compute their subgroup key.
$ssid_i = ((U_1, y_1), \ldots, (U_M, y_M))$ containing all $U_i \in \mathbf{S}$ and their y_i taken from GKE Stage.

U_{i-1} 

$z_{i-1} = H(y_i^{x_{i-1}}, ssid_{i-1}) \oplus H(y_{i-2}^{x_{i-1}}, ssid_{i-1})$

U_i 

$z_i = H(y_{i+1}^{x_i}, ssid_i) \oplus H(y_{i-1}^{x_i}, ssid_i)$

U_{i+1} 

$z_{i+1} = H(y_{i+2}^{x_{i+1}}, ssid_{i+1}) \oplus H(y_i^{x_{i+1}}, ssid_{i+1})$

$$z_1 \oplus \cdots \oplus z_M \stackrel{?}{=} 0\ ;\ \text{recover } z_{1,2}, \ldots, z_{M,1}$$

$$\text{subgroup key } k_{i,S} := H(z_{1,2}, \ldots, z_{M,1}, (U_1, y_1), \ldots, (U_M, y_M))$$

Authentication and Performance

Authentication

Our mBD+P and mBD+S protocols use signature-based authentication [KaYu03].

In mBD+P and mBD+S (GKE Stage) signature $\sigma_i = \mathrm{Sign}(sk_i, (U_i, z_i, sid_i))$

In mBD+S (Subgroup Stage) signature $\sigma_i = \mathrm{Sign}(sk_i, (U_i, z_i, ssid_i))$

Performace

Comparison with protocols from [Man09], excluding authentication costs:

GKE+P/S	Rounds	Communication (in log Q bits)	Computation (in mod. exp. per U_i)
GKE+P BD[Man09]	2	3N	3
GKE+P KPT[Man09]	2	2N – 2	N + 2 – i (2N – 2 for U_i)
mBD+P	2	2N	3
GKE+S BD (Subgroup Stage)	2	2M	2
mBD+S (Subgroup Stage)	1	M	≤ 2 (via tade-off)

Conclusion

Flexible Group Key Exchange ⇒ 1 group key + multiple subgroup/p2p keys

GKE+S as a general case of GKE+P from [Man09]

New security challenges

Independence between group key, subgroup keys, and p2p keys.

Consideration of insider and collusion attacks.

Constructions

Modified BD protocol to allow re-use of exponents x_i for the computation of all keys.

mBD+P for *non-interactive* derivation of p2p keys (more efficient than in [Man09]).

mBD+S as extension of mBD+P for *efficient* computation of subgroup keys *(1 round).*

Not in the talk

Security model for GKE+S protocols as extension of [KaYu03] model and *proofs*.