Tightly-Secure Group Key Exchange with Perfect Forward Secrecy

Emanuele Di Giandomenico*, Doreen Riepel, Sven Schäge

ASIACRYPT 2024
13th December 2024

Introduction

What is a Group Authenticated Key Exchange (GAKE)?

Introduction

What is a Group Authenticated Key Exchange (GAKE)?

- Generalizes two-party key exchange to a group setting.

Introduction

What is a Group Authenticated Key Exchange (GAKE)?

- Generalizes two-party key exchange to a group setting.
- Enables secure symmetric session keys for group communication.

Introduction

What is a Group Authenticated Key Exchange (GAKE)?

- Generalizes two-party key exchange to a group setting.
- Enables secure symmetric session keys for group communication.

Why GAKE matters:

Introduction

What is a Group Authenticated Key Exchange (GAKE)?

- Generalizes two-party key exchange to a group setting.
- Enables secure symmetric session keys for group communication.

Why GAKE matters:

- Essential for secure group communication over insecure networks.

Introduction

What is a Group Authenticated Key Exchange (GAKE)?

- Generalizes two-party key exchange to a group setting.
- Enables secure symmetric session keys for group communication.

Why GAKE matters:

- Essential for secure group communication over insecure networks.

P_i $(\mathsf{pk}_i, \mathsf{sk}_i)$		$\mathcal{P}_i$
$(m_i, \mathsf{st}) \leftarrow \mathsf{Begin}(\mathsf{sk}_i, \{\mathsf{pk}_j\}_{j \in \mathcal{P}_i})$	$\xrightarrow{(i, m_i)}$	
	$\xleftarrow{\mathcal{M}_i = \{(j, m_j)\}_{j \in \mathcal{P}_i}}$	
$(\hat{m}_i, \mathsf{st}) \leftarrow \mathsf{Respond}(\mathsf{sk}_i, \mathsf{st}, \mathcal{M})$	$\xrightarrow{(i, \hat{m}_i)}$	
	$\xleftarrow{\hat{\mathcal{M}}_i = \{(j, \hat{m}_j)\}_{j \in \mathcal{P}_i}}$	
$(\bar{m}_i, \mathsf{st}) \leftarrow \mathsf{Final}(\mathsf{sk}_i, \mathsf{st}, \hat{\mathcal{M}})$	$\xrightarrow{(i, \bar{m}_i)}$	
	$\xleftarrow{\bar{\mathcal{M}}_i = \{(j, \bar{m}_j)\}_{j \in \mathcal{P}_i}}$	
$K := \mathsf{Derive}(\mathsf{sk}_i, \mathsf{st}, \bar{\mathcal{M}})$		

Challenges in GAKE

Quantum Attacks:

Challenges in GAKE

Quantum Attacks:

- Vulnerable due to reliance on classical security assumptions.

Challenges in GAKE

Quantum Attacks:

- Vulnerable due to reliance on classical security assumptions.

Weak Security Models:

Challenges in GAKE

Quantum Attacks:

- Vulnerable due to reliance on classical security assumptions.

Weak Security Models:

- Many protocols ignore Maximum Exposure Attacks (MEX).

Challenges in GAKE

Quantum Attacks:

- Vulnerable due to reliance on classical security assumptions.

Weak Security Models:

- Many protocols ignore Maximum Exposure Attacks (MEX).

Non-Tight Security Proofs:

Challenges in GAKE

Quantum Attacks:

- Vulnerable due to reliance on classical security assumptions.

Weak Security Models:

- Many protocols ignore Maximum Exposure Attacks (MEX).

Non-Tight Security Proofs:

- Inefficient parameter settings due to loose reductions.

Burmester-Desmedt Protocol [BD95]

Let p be a prime number and g an element of a group G. Let n be the number of parties.

Burmester-Desmedt Protocol [BD95]

Let p be a prime number and g an element of a group G. Let n be the number of parties.

- Each P_i selects x_i and broadcasts $k_i = g^{x_i} \mod p$.

Burmester-Desmedt Protocol [BD95]

Let p be a prime number and g an element of a group G. Let n be the number of parties.

- Each P_i selects x_i and broadcasts $k_i = g^{x_i} \mod p$.
- Each P_i broadcasts $c_i = (k_{i+1}/k_{i-1})^{x_i} \mod p$.

Burmester-Desmedt Protocol [BD95]

Let p be a prime number and g an element of a group G. Let n be the number of parties.

- Each P_i selects x_i and broadcasts $k_i = g^{x_i} \mod p$.
- Each P_i broadcasts $c_i = (k_{i+1}/k_{i-1})^{x_i} \mod p$.
- Each P_i computes the key $K = k_{i-1}^{nx} \cdot c_i^{n-1} \cdot c_{i+1}^{n-2} \cdots c_{i-2} \mod p$ $= g^{x_1 x_2 + x_2 x_3 + \cdots + x_n x_1} \mod p$.

Burmester-Desmedt Protocol [BD95]

Let p be a prime number and g an element of a group G. Let n be the number of parties.

- Each P_i selects x_i and broadcasts $k_i = g^{x_i} \mod p$.
- Each P_i broadcasts $c_i = (k_{i+1}/k_{i-1})^{x_i} \mod p$.
- Each P_i computes the key $K = k_{i-1}^{nx} \cdot c_i^{n-1} \cdot c_{i+1}^{n-2} \cdots c_{i-2} \mod p$ $= g^{x_1 x_2 + x_2 x_3 + \cdots + x_n x_1} \mod p$.

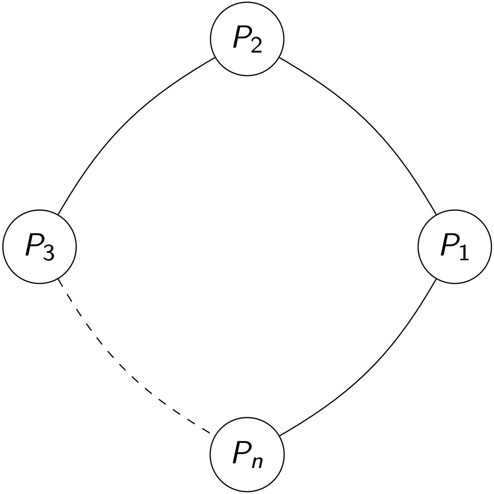
Using digital signatures over all messages sent, this protocol can be made actively secure.

Fresh Perspective on BD Protocol

A ring structure for the parties.

Fresh Perspective on BD Protocol

A ring structure for the parties.



Fresh Perspective on BD Protocol

A ring structure for the parties.
First phase:

Fresh Perspective on BD Protocol

A ring structure for the parties.
First phase: adjacent parties compute a common session key via a two-party protocol.

Fresh Perspective on BD Protocol

A ring structure for the parties.
First phase: adjacent parties compute a common session key via a two-party protocol.

- P_i authenticates P_{i-1} and P_{i+1}, sending k_i and digital signature.

Fresh Perspective on BD Protocol

A ring structure for the parties.
First phase: adjacent parties compute a common session key via a two-party protocol.

- P_i authenticates P_{i-1} and P_{i+1}, sending k_i and digital signature.
- The shared key with party P_{i+1} is $K_{i,i+1} = (k_{i+1})^{x_i}$.

Fresh Perspective on BD Protocol

A ring structure for the parties.
First phase: adjacent parties compute a common session key via a two-party protocol.

- P_i authenticates P_{i-1} and P_{i+1}, sending k_i and digital signature.
- The shared key with party P_{i+1} is $K_{i,i+1} = (k_{i+1})^{x_i}$.
- The shared key with P_{i-1} is $K_{i-1,i} = (k_{i-1})^{x_i}$.

Fresh Perspective on BD Protocol

A ring structure for the parties.
First phase: adjacent parties compute a common session key via a two-party protocol.

- P_i authenticates P_{i-1} and P_{i+1}, sending k_i and digital signature.
- The shared key with party P_{i+1} is $K_{i,i+1} = (k_{i+1})^{x_i}$.
- The shared key with P_{i-1} is $K_{i-1,i} = (k_{i-1})^{x_i}$.

Second phase:

Fresh Perspective on BD Protocol

A ring structure for the parties.
First phase: adjacent parties compute a common session key via a two-party protocol.

- P_i authenticates P_{i-1} and P_{i+1}, sending k_i and digital signature.
- The shared key with party P_{i+1} is $K_{i,i+1} = (k_{i+1})^{x_i}$.
- The shared key with P_{i-1} is $K_{i-1,i} = (k_{i-1})^{x_i}$.

Second phase: distribute the derived keys to the authenticated parties.

Fresh Perspective on BD Protocol

A ring structure for the parties.

First phase: adjacent parties compute a common session key via a two-party protocol.

- P_i authenticates P_{i-1} and P_{i+1}, sending k_i and digital signature.
- The shared key with party P_{i+1} is $K_{i,i+1} = (k_{i+1})^{x_i}$.
- The shared key with P_{i-1} is $K_{i-1,i} = (k_{i-1})^{x_i}$.

Second phase: distribute the derived keys to the authenticated parties.

- $K_{i,i+1}$ is only given to P_{i-1} and $K_{i-1,i}$ is only given to P_{i+1}, then publish $K_{i,i+1}/K_{i-1,i} = (k_{i+1}/k_{i-1})^{x_i} = c_i$.

Fresh Perspective on BD Protocol

A ring structure for the parties.
First phase: adjacent parties compute a common session key via a two-party protocol.

- P_i authenticates P_{i-1} and P_{i+1}, sending k_i and digital signature.
- The shared key with party P_{i+1} is $K_{i,i+1} = (k_{i+1})^{x_i}$.
- The shared key with P_{i-1} is $K_{i-1,i} = (k_{i-1})^{x_i}$.

Second phase: distribute the derived keys to the authenticated parties.

- $K_{i,i+1}$ is only given to P_{i-1} and $K_{i-1,i}$ is only given to P_{i+1}, then publish $K_{i,i+1}/K_{i-1,i} = (k_{i+1}/k_{i-1})^{x_i} = c_i$.
- P_i can compute $K = g^{x_1x_2+x_2x_3+\ldots+x_nx_1} = K_{1,2}K_{2,3}\cdots K_{n,1}$.

Fresh Perspective on BD Protocol

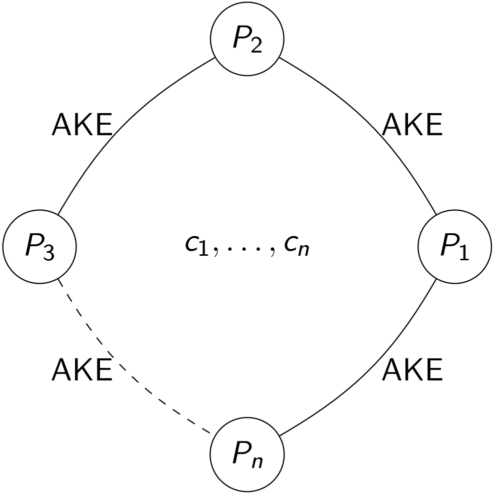
A ring structure for the parties.
First phase: adjacent parties compute a common session key via a two-party protocol.

- P_i authenticates P_{i-1} and P_{i+1}, sending k_i and digital signature.
- The shared key with party P_{i+1} is $K_{i,i+1} = (k_{i+1})^{x_i}$.
- The shared key with P_{i-1} is $K_{i-1,i} = (k_{i-1})^{x_i}$.

Second phase: distribute the derived keys to the authenticated parties.

- $K_{i,i+1}$ is only given to P_{i-1} and $K_{i-1,i}$ is only given to P_{i+1}, then publish $K_{i,i+1}/K_{i-1,i} = (k_{i+1}/k_{i-1})^{x_i} = c_i$.
- P_i can compute $K = g^{x_1x_2+x_2x_3+\ldots+x_nx_1} = K_{1,2}K_{2,3}\cdots K_{n,1}$. It can step-wisely compute the value
  $K_{i,i+1} = K_{i-1,i} \cdot c_i \iff K_{i,i+1}/c_i = K_{i-1,i}$.

Fresh Perspective on BD Protocol



$$c_i = K_{i,i+1}/K_{i-1,i} = SymEnc(K_{i,i+1}, K_{i-1,i}) = SymEnc'(K_{i-1,i}, K_{i,i+1}).$$

Novel Concept

- P_i authenticates P_{i-1} and vice versa is not required, only one direction is enough.

Novel Concept

- P_i authenticates P_{i-1} and vice versa is not required, only one direction is enough.
- Use Unilateral Authenticated Key Exchange (UAKE) instead of AKE.

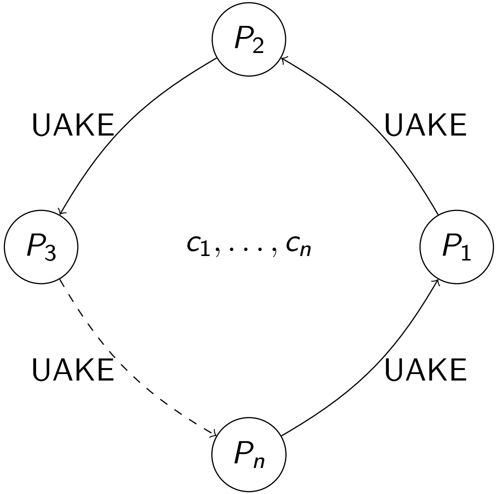
Novel Concept

- P_i authenticates P_{i-1} and vice versa is not required, only one direction is enough.
- Use Unilateral Authenticated Key Exchange (UAKE) instead of AKE.
- In the second phase, P_i distributes the symmetric key shared with P_{i+1} to P_{i-1} (and not vice versa).

Novel Concept

- P_i authenticates P_{i-1} and vice versa is not required, only one direction is enough.
- Use Unilateral Authenticated Key Exchange (UAKE) instead of AKE.
- In the second phase, P_i distributes the symmetric key shared with P_{i+1} to P_{i-1} (and not vice versa).
- A random oracle-based symmetric encryption system is used, where the sharing of key $K_{i,i+1}$ to P_{i-1} now proceeds as $c_i = h(K_{i-1,i}) \oplus K_{i,i+1}$.

Novel Concept



$$c_i = h(K_{i-1,i}) \oplus K_{i,i+1} = SymEnc(K_{i,i+1}, K_{i-1,i})$$

Design Choices

One-way Authentication:

- Each party authenticates its predecessor, saving resources.

Design Choices

One-way Authentication:

- Each party authenticates its predecessor, saving resources.

KEM-Based Authentication:

- Replaces traditional signatures with efficient encapsulation mechanisms.

Design Choices

One-way Authentication:

- Each party authenticates its predecessor, saving resources.

KEM-Based Authentication:

- Replaces traditional signatures with efficient encapsulation mechanisms.

Random Oracle Model (ROM):

- Utilized for tight security guarantees.

Transformations

KEM to UAKE$_{\mathsf{WFS}}$:

- Utilize an ephemeral public key and key encapsulation to derive a fresh session key, ensuring authenticated encryption tied to the long-term key.

Transformations

KEM to $\mathrm{UAKE}_{\mathrm{WFS}}$:

- Utilize an ephemeral public key and key encapsulation to derive a fresh session key, ensuring authenticated encryption tied to the long-term key.

Weak to (Full) Perfect Forward Secrecy:

- Introduce key confirmation since it is unilaterally authenticated, and it does not increase number of moves.

Transformations

KEM to UAKE_{WFS}:

- Utilize an ephemeral public key and key encapsulation to derive a fresh session key, ensuring authenticated encryption tied to the long-term key.

Weak to (Full) Perfect Forward Secrecy:

- Introduce key confirmation since it is unilaterally authenticated, and it does not increase number of moves.

UAKE to GAKE Transformation:

- An approach for extending bilateral protocols to the group setting.

Transformations

KEM to $\mathsf{UAKE}_{\mathsf{WFS}}$:

- Utilize an ephemeral public key and key encapsulation to derive a fresh session key, ensuring authenticated encryption tied to the long-term key.

Weak to (Full) Perfect Forward Secrecy:

- Introduce key confirmation since it is unilaterally authenticated, and it does not increase number of moves.

UAKE to GAKE Transformation:

- An approach for extending bilateral protocols to the group setting.

$$\mathsf{KEM} \longrightarrow \mathsf{UAKE}_{\mathsf{WFS}} \longrightarrow \mathsf{UAKE}_{\mathsf{PFS}} \longrightarrow \mathsf{GAKE}$$

Transformations

KEM to UAKE_{WFS}:

- Utilize an ephemeral public key and key encapsulation to derive a fresh session key, ensuring authenticated encryption tied to the long-term key.

Weak to (Full) Perfect Forward Secrecy:

- Introduce key confirmation since it is unilaterally authenticated, and it does not increase number of moves.

UAKE to GAKE Transformation:

- An approach for extending bilateral protocols to the group setting.

$$
\begin{tikzcd}
\text{DDH} \arrow[dr] & & & \\
& \text{KEM} \arrow[r] & \text{UAKE}_{\text{WFS}} \arrow[r] & \text{UAKE}_{\text{PFS}} \arrow[r] & \text{GAKE}
\end{tikzcd}
$$

Transformations

KEM to $\mathsf{UAKE}_{\mathsf{WFS}}$:

- Utilize an ephemeral public key and key encapsulation to derive a fresh session key, ensuring authenticated encryption tied to the long-term key.

Weak to (Full) Perfect Forward Secrecy:

- Introduce key confirmation since it is unilaterally authenticated, and it does not increase number of moves.

UAKE to GAKE Transformation:

- An approach for extending bilateral protocols to the group setting.

$$\begin{tikzcd}
\mathsf{DDH} \arrow[dr] & & & & \\
& \mathsf{KEM} \arrow[r] & \mathsf{UAKE}_{\mathsf{WFS}} \arrow[r] & \mathsf{UAKE}_{\mathsf{PFS}} \arrow[r] & \mathsf{GAKE} \\
\mathsf{LWE} \arrow[ur, "\text{[PWZ23]}"'] & & & &
\end{tikzcd}$$

Security Features

A new conceptual approach that re-uses two-party definitions:

- We use [JKRS21] strong security definition as a starting point.

Security Features

A new conceptual approach that re-uses two-party definitions:

- We use [JKRS21] strong security definition as a starting point.

Maximum Exposure Attack Resistance:

- Secure even when attackers reveal session states or long-term keys.

Security Features

A new conceptual approach that re-uses two-party definitions:

- We use [JKRS21] strong security definition as a starting point.

Maximum Exposure Attack Resistance:

- Secure even when attackers reveal session states or long-term keys.

Tight Security Proofs:

- Independent of group size, sessions, or adversary queries.

Security Features

A new conceptual approach that re-uses two-party definitions:

- We use [JKRS21] strong security definition as a starting point.

Maximum Exposure Attack Resistance:

- Secure even when attackers reveal session states or long-term keys.

Tight Security Proofs:

- Independent of group size, sessions, or adversary queries.

(Full) Perfect Forward Secrecy:

- Allows group session keys to remain secure even if long-term secret keys are later compromised.

Security Features

A new conceptual approach that re-uses two-party definitions:

- We use [JKRS21] strong security definition as a starting point.

Maximum Exposure Attack Resistance:

- Secure even when attackers reveal session states or long-term keys.

Tight Security Proofs:

- Independent of group size, sessions, or adversary queries.

(Full) Perfect Forward Secrecy:

- Allows group session keys to remain secure even if long-term secret keys are later compromised.

Post-Quantum Readiness:

- Lattice-based constructions secure against quantum adversaries.

Conclusion

- A new conceptual approach to build GAKE.

Conclusion

- A new conceptual approach to build GAKE.
- A GAKE based on KEMs both for exchange and authentication.

Conclusion

- A new conceptual approach to build GAKE.
- A GAKE based on KEMs both for exchange and authentication.
- Post-quantum tight security proof in a security model where reveal state queries are allowed.

Conclusion

- A new conceptual approach to build GAKE.
- A GAKE based on KEMs both for exchange and authentication.
- Post-quantum tight security proof in a security model where reveal state queries are allowed.
- Perfect Forward Secrecy is guaranteed.

Questions?

Thanks for your attention!