Tampere University

Arrows in A Quiver: A Secure Certificateless Group Key Distribution Protocol for Drones

Authors: Eugene Frimpong
Reyhaneh Rabbaninejad
Antonis Michalas

Outline

- Overview of Arrows in a Quiver (AinQ)
- System Model
- Proposed Scheme
- Security Analysis
- Proposed Protocol
- Experimental Results
- Questions

Overview

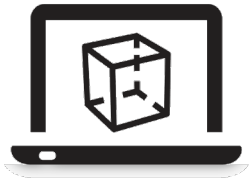
- A group key distribution scheme and protocol designed for a Drone ecosystem.
- Supports drone mobility (i.e., drones leaving and joining a set group)

Points to Consider

- ✓ Architecture Model – Centralized or Distributed – Key Agreement or Key Distribution
- ✓ Key Distribution Model – Synchronous or Asynchronous
- ✓ Reliability – Group key consistency
- ✓ Protocol Overhead – Computational and message overhead

Acknowledgment

Work supported by the Technology Innovation Institute (TII), Abu Dhabi, for the project ARROWSMITH: Living (Securely) on the Edge



System Model

Cloud Service Provider (CSP): An abstract external entity which acts as the destination for all collected data. Specifics out of scope of this work.

Key Generation Center (KGC): This is the trusted entity responsible for generating and setting the system parameters for the complete run of the protocol. The KGC generates partial private and public keys for each entity during the Protocol Setup Phase.

Edge Drones: Let $D = \{d_1, \dots, d_n\}$ be the set of all drones in our environment. Each drone is equipped with several sensors to monitor and report on sensed events. We use Saml11-xpro and Zolertia Re-mote Revb boards for our experiments.

Drone Team Leader: Let $\mathrm{Q} = \{q_1, \dots, q_m\}$ be the set of all drones elected as Group Leaders in our protocol. Each drone team leader maintains a list of all drones in its group and updates this list whenever a new drone joins the group, or an existing member exists the group. We use an UP Xtreme Intel i7 board for our experiments.

Proposed Scheme

- Based on pairing-free Certificateless Public-Key Cryptography. The proposed scheme uses standard ECC primitives.

- AinQ is a tuple of 8 algorithms:
  - *SetUp*
  - *GenSecretValue*
  - *GenPartialKey*
  - *FullKeyGen*
  - *GenGroupKey*
  - *KeyRetrieval*
  - *Re-Key*

Setup

$\lambda \in \mathbb{Z}^+$

q, P, G_q

KGC

$x \in \mathbb{Z}^*$

$P_{pub} = xP$

$\Omega = \{F_q, \frac{E}{F_q}, G_q, P, P_{pub}, H_0, H_1\}$

GenSecretValue

Team Leader q_k

- $x_k \in \mathbb{Z}_q^*$
- $P_k = x_k P$

Drone 1 d_i

- $x_i \in \mathbb{Z}_q^*$
- $P_i = x_i P$

Drone 2 d_j

- $x_j \in \mathbb{Z}_q^*$
- $P_j = x_j P$

GenPartialKey

(d_i, P_i)

- $r_i \in \mathbb{Z}^*$
- $R_i = r_i \,.\, P$
- $s_i = r_i + xH_0(d_i, R_i, P_i)\ mod\ q$

s_i, R_i

FullKeyGen

Team Leader q_k

- $sk_k = (x_k, s_k)$
- $pk_k = (P_k, R_k)$

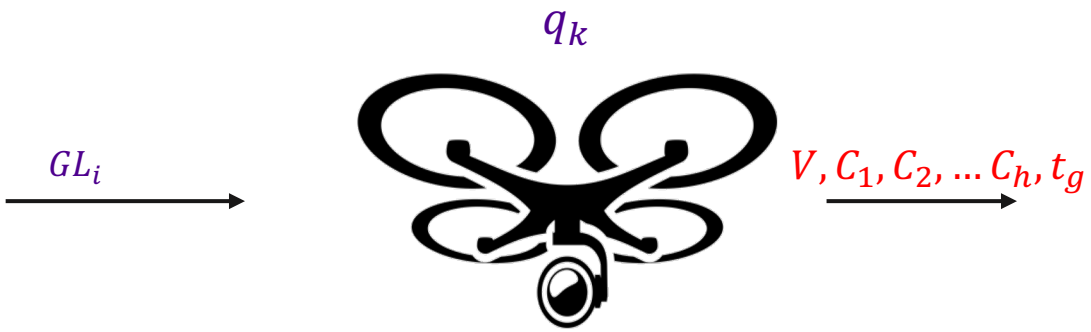
Drone 1 d_i

- $sk_i = (x_i, s_i)$
- $pk_i = (P_i, R_i)$

Drone 2 d_j

- $sk_j = (x_j, s_j)$
- $pk_j = (P_j, R_j)$

GenGroupKey



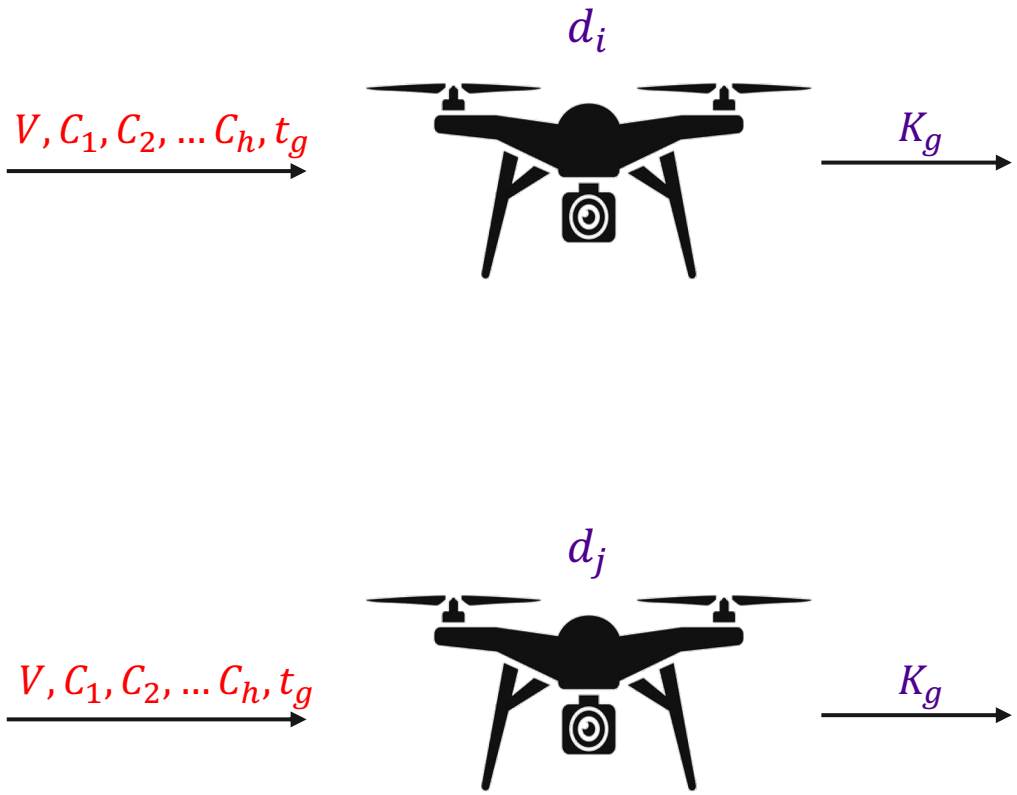
Given a group list GL_i

The team leader:

- choose $K_g, l_k \in \mathbb{Z}_q^*$
- $V = l_k P$
- For each edge drone
  - $Y_i = R_i + H_0(d_i, R_i, P_i) \cdot P_{pub} + P_i$
  - $T_i = l_k \cdot Y_i$
  - $C_i = K_g \oplus H_1(V, T_i, q_k, pk_k, d_i, pk_i, t_g)$

Where GL_i contains a list of group members $\{d_1, \dots, d_h\}$ and their respective public keys $pk_i, i \in \{1, \dots, h\}$

KeyRetrieval



Each edge drone retrieves K_g:

- $T_i = (x_i + s_i) . V$
  $(x_i + s_i) . l_k . P = l_k . Y_i$
- $K_g = C_i \oplus H_1(V, T_i, q_k, pk_k, d_i, pk_i, t_g)$

Re-Keying

Given the updated group list GL_i'

The team leader:

- choose $K_g' \in \mathbb{Z}_q^*$
- For every new edge drone
  - $Y_i = R_i + H_0(d_i, R_i, P_i) \cdot P_{pub} + P_i$
  - $T_i = l_k \cdot Y_i$
- For all drones:
  - $C_i' = K_g' \oplus H_1(V, T_i, q_k, pk_k, d_i, pk_i, t_g)$

Where GL_i' contains an updated list of group members $\{d_1, \ldots, d_g\}$ and their respective public keys $pk_i, i \in \{1, \ldots, g\}$

Security Analysis

We prove the security of our GKD scheme in the presence of a malicious adversary A who can be an

- outside adversary– a passive eavesdropper or a malicious entity who has captured some drones
- inside adversary– a corrupt KGC and a revoked user

We formally define **Ainq-IND-CCA2 Game**

- Query phase- A adaptively queries $O_{GenSecretValue}$ $O_{GenPartialKey}$ $O_{GenGroupKey}$ $O_{KeyRetrieval}$ O_{Re-Key} oracles
- Challenge phase
  - A submits two distinct chosen session keys K_0, K_1
  - Challenger selects a random bit b ∈ {0, 1}, and sends the "challenge" ciphertext (encryption of K_b)
  - Finally, after running another query phase, the adversary outputs a guess for the value of b.

We prove AinQ is IND-CCA2 secure under DDH assumption in the random oracle model by formally proving that no adversary has a non-negligible advantage in winning the Ainq-IND-CCA2 Game.

Security Analysis

Following the formal security proof, AinQ satisfies the following security requirements of a GKD scheme

- Key Freshness: trivially satisfied since each new session key is chosen uniformly at random from the key space making the event of repetitious session keys unlikely to happen.
- Group Key Secrecy: is a trivial inference of IND-CCA2 security.
- Forward/Backward Secrecy: whenever a new revocation/enrolment happens, the session key is refreshed by executing Re − Key algorithm, so newly revoked/enrolled drone's view is exactly same as the view of an outside adversary, and AinQ satisfies these properties based on its IND-CCA2 security.

Proposed Protocol

The proposed protocol is divided into three phases;

- Setup and Initialization
- Key Generation and Retrieval
- Group Re-Key

To provide a detailed description of each phase, we consider a scenario consisting of an elected drone group/team leader and several regular drones in its group.

In our assumed scenario, the elected drone team leader $\boldsymbol{q_k}$ wishes to generate a group key $\boldsymbol{K_g}$ for its group in the presence of a trusted **KGC**. The list of group members is denoted by $\boldsymbol{GL_i = \{d_1, d_2, \ldots, d_h\}}$

Key Generation and Retrieval

q_k

d_j

- Generate $\boldsymbol{K_g}, \boldsymbol{V}$
- Compute $\boldsymbol{C_1}, \boldsymbol{C_2}, \ldots, \boldsymbol{C_h}$
- Compute $\boldsymbol{m_1} = (\boldsymbol{r_1}, \boldsymbol{V}, \boldsymbol{C_1}, \boldsymbol{C_2}, \ldots, \boldsymbol{C_h}, \boldsymbol{q_k}, \boldsymbol{t_g}, \boldsymbol{\sigma_{qk}})$

$\boldsymbol{m_1}$

- Compute $\boldsymbol{T_i}$
- Compute $\boldsymbol{K_g}$
- Verify $\boldsymbol{m_1}$

where $\boldsymbol{\sigma_{qk}} = \boldsymbol{sig_{q_k}}(\boldsymbol{r_1}||\boldsymbol{V}||\boldsymbol{K_g})$

Experimental Results

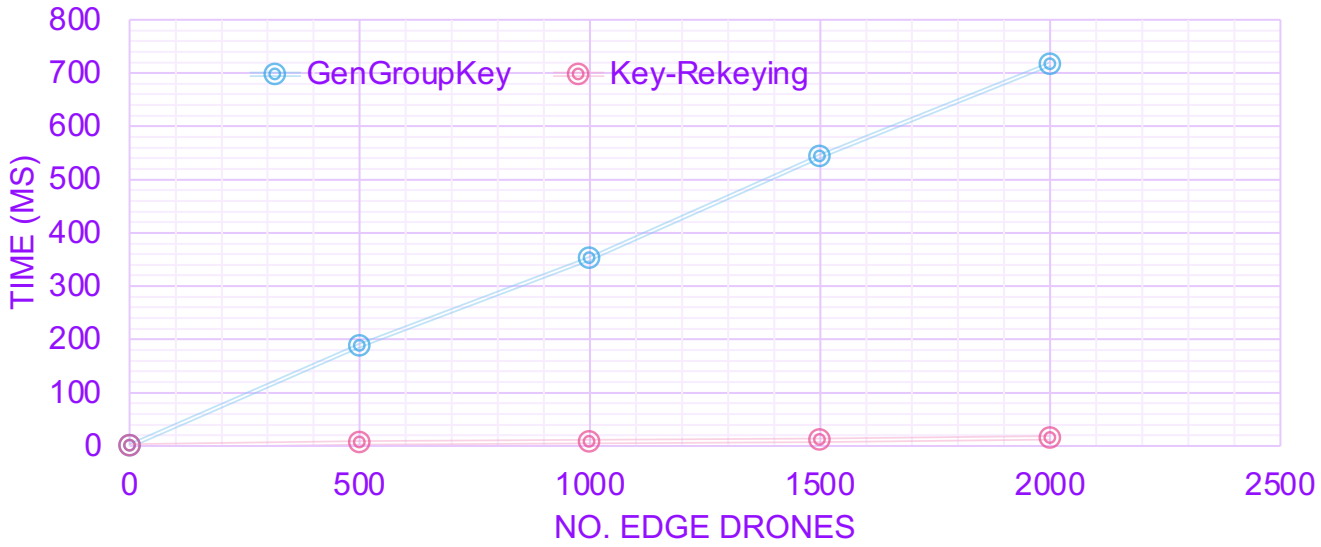
	SAML11-Xpro	Zolertia Re-Mote
SoC	ARM Cortex-M23	ARM Cortex-M3
RAM	16KB	32KB
Flash	64KB	512KB
Clock Speed	32MHz	32MHz

		SAML11-Xpro	Zolertia Re-Mote
	EC Multiplication	Time (sec)	Time (sec)
EC Multiplication	0	4.782	2.598
GenSecretKey	1	5.343	2.943
KeyRetrieval	1	4.783	2.613

Experimental Results

q_k

	UP Xtreme
SoC	Intel i7-8665UE
RAM	16GB
Flash	64GB
Graphics	Intel UHD 620



All source code publicly available at https://github.com/iammrgenie/AinQ

Questions