Given S, a finite set of security groups, the language our mechanism operates on can be generated by a context free grammar. Because the language is dependent on the security groups S, this grammar must be generated based on it. This is done in two steps:

First, we define the base grammar:

$$
\begin{array}{lll}
G^1 = (V^1, \Sigma^1, R^1, \mathcal{A}), \text{ where} & & \\
V^1 = \{\mathcal{A}, W\} & & \text{non-terminal symbols} \\
\Sigma^1 = \{\emptyset\} & & \text{terminal symbols} \\
R^1 = \{ & \mathcal{A} \to \varepsilon, & \text{rules of production} \\
 & \mathcal{A} \to W\mathcal{A} | W\} &
\end{array}
$$

This base grammar, through the non-terminal symbols and production rules, establishes a means of generating the base language form of unordered windows ($W \in V^1$) in an arbitrary length such as WW or $WWWWW$.

Next, we generate the S specific definitions. To do so it is first necessary to define notation for two special terminal symbols and three special sets:

$o_{s,i}$ - an open command issued to element i within security group s,
$a_{s,i}$ - an acknowledgement received from element i within security group s,
θ_s - the set of all $o_{s,i}$ terminals for security group s,
α_s - the set of all $a_{s,i}$ terminals for security group s, and
$\pi(A)$ - the set of all permutations of the set A.

These definitions allow us to define a final, special set:

$$\Lambda_s = \pi(\theta_s) \times \pi(\alpha_s)$$

Intuitively, Λ_s is a set of ordered sets expressing each permutation of the θ_s set matched with each permutation of the α_s set. For example, given a security group s made up of two elements s.t. $s = \{1, 2\}$, Λ_s is defined:

$$
\begin{array}{ll}
\Lambda_s = \{(o_{s,1}, o_{s,2}, a_{s,1}, a_{s,2}), & (o_{s,2}, o_{s,1}, a_{s,1}, a_{s,2}), \\
\quad (o_{s,1}, o_{s,2}, a_{s,2}, a_{s,1}), & (o_{s,2}, o_{s,1}, a_{s,2}, a_{s,1})\}
\end{array}
$$

The sets within Λ_s represent all legitimate command sequences within a window (W) for security group s. A key property of the sets within Λ_s is that each element within the security group is issued an open command, in any order, followed by acknowledgements from each element within the security group, once again in any order.

With these definitions established we can now formally define an S specific grammar:

$$
\begin{array}{l}
G^2 = (V^2, \Sigma^2, R^2, \emptyset), \text{ where} \\
V^2 = \{W\} \\
\Sigma^2 = \{[o_{s,i}, a_{s,i}] : \forall i \in \forall s \in S\} \\
R^2 = \{[W \to \lambda] : \forall \lambda \in \Lambda_s : \forall s \in S\}
\end{array}
$$

These definitions add new terminal symbols and the necessary production rules to generate them.

The production rules are slightly more complex to generate. For every security group $s \in S$

With these symbols and the production rule $W_s \rightarrow o_s W_s a_s$ the grammar is now capable of filling the windows W with open and acknowledgement messages that allow a single security group to communicate within a windows.

Finally, the language our mechanism accepts for security group S can be formed using the union of the previous two grammars:

$G = (V, \Sigma, R, \mathcal{A})$, where
$V = V^1 \cup V^2$
$\Sigma = \Sigma^1 \cup \Sigma^2$
$R = R^1 \cup R^2$