A Symbols and Keywords

Symbol	Description	Example
`.`	associative concatenation (of messages)	`SND(ABC.XY.Z)`
`,`	separates elements of a set, or arguments of a predicate or role	
`'`	prime, used for referring to the next (new) value of variable in a transition	`X'`
`;`	sequential composition of roles	`Phase1(...);` `Phase2(...)`
`%`	comment (until end of line)	
`:=`	initialisation (of local variable) in init-section	`init X := 1`
`:=`	assignment to (primed!) local variable	`X' := 1`
`=`	equality test of assigned variables or other expressions	`X = 1`
`<`	less than	`X < 2`
`/\`	conjunction (logical AND)	`X = 2 /\ Y = 3`
`/\`	parallel composition of roles	`alice(...) /\ bob(...)`
`/\_`	conjunction over elements in a set	`/\_{in(A,Agents)}` `Kr(A)=[]`
`->`	mapping from one data-type to another	`KeyMap: agent -> public_key`
`=|>`	immediate transition	`RCV(X) =|> SND(Y)`
`{ }`	set delimiter e.g. in knowledge declaration	`{a,b}`
`{ }_`	encryption or signature	`SND({X}_K)`
`( )`	indicates arguments of function, send or receive statement, or role.	
`accept`	used in sequential composition to indicate when a role is finished and the new role can begin	`accept` `State=5 /\ Auth=1`
`agent`	data-type for agents	
`bool`	data-type for boolean values	
`channel(dy)`	data-type for channels. Currently only Dolev-Yao channels implemented.	
`composition`	marks beginning of composition section of a composed role	
`cons`	add element to set	`L' = cons(X,L)`

Symbol	Description	Example
`def=`	indicates beginning of body of a role	
`delete`	delete element from set	`L' = delete(X,L)`
`end role`	indicates end of role	
`exp`	exponentiation operator (prefix)	`exp(g,x)` represents g^x
`hash_func`	data-type for one-way functions	
`i`	intruder's identity	
`in`	check if element is in list or set	`in(X,L)`
`init`	indicates initialisation of local variables	`init State := 0`
`inv`	inverse of a key: given a public key returns private key	`inv(Ka)`
`intruder_knowledge`	defines knowledge of the intruder	`intruder_knowledge={a,kai}`
`local`	indicates local variable section	`local State : nat`
`message`	general type of message contents	
`nat`	data-type for natural numbers	
`not`	logical negation	`not(in(X,L))`
`owns`	ownership of a variable: if a role owns a variable, only this role may change the value of the variable	`owns X`
`played_by`	for basic roles: specifies which agent is playing this role	`played_by A`
`public_key`	data-type for public keys	
`request`	used to check strong authentication (together with `witness`)	`request(A,B, alice_bob_na,Na)`
`secret`	used to check secrecy	`secret(K,k,{A,B})`
`set`	data-type for unordered collection of typed values	`local S : text set` `init S := {}`
`symmetric_key`	data-type for symmetric keys	
`text`	data-type for uninterpreted bit-strings (like nonces)	
`transition`	marks beginning of transitions section of basic role	
`witness`	used to check authentication (together with `(w)request`)	`witness(B,A, bob_alice_na,Na)`
`wrequest`	used to check weak authentication (together with `witness`)	`wrequest(A,B, alice_bob_na,Na)`
`xor`	prefix xor operator	`xor(a,b)`