appendix2

1. The modeling and implementation process

Scyther, as a formal verification tool, doesn't directly provide formal verification for authenticity. It necessitates verifying its authenticity through secret, aliveness, weak agreement, non-injective agreement, and non-injective synchronization. the strength of authenticity increases sequentially for Aliveness, weak agreement, non-injective agreement, and non-injective synchronization. Aliveness represents liveness authentication, serving as a fundamental form of authentication that ensures the expected communicating party A exists. Weak-agree, denoting weak protocol authentication, demands that certain states or values among participating parties remain consistent throughout the protocol's execution. Niagree, referring to non-monotonic agreement authentication, describes how communication or negotiation outcomes between parties cannot be repudiated during protocol execution. Nisynch, or non-injective synchronization authentication, indicates that, in the event an attacker gains access to the private key of proxy A, all send/receive events preceding a claimed event are correctly executed by proxy A in the correct order and content. This property ensures the integrity of received messages, Nisynch's definition is quite similar to Niargree's, with the distinction that Nisynch adds a requirement for expected order, thus exhibiting stronger authenticity.

The modeling and implementation process is shown in Table 5:

Table 1: The modeling and implementation process in scyther

1. A_1 performs computational operations.	match $(e, H(R, PK, m))$; match $(TIDa, H(A_1, g))$; match $(w, H(TIDa, pw))$; match $(m, H(TIDa, TIDdb, w, T1, NA1, pk(A_1), QA, e, y))$
2. A_1 sends a message to DB.	$send_2(A_1, DB, TIDa, DB, w, T1, NA1, pk(A_1), y, m, QApk(DB))$;
3. DB receives the message.	$recv_2(A_1, DB, TIDa, DB, w, T1, NA1, pk(A1), y, m, QApk(DB))$;
4. DB sends a signature conversion request to the inter-chain notary NP_1	$send_3(DB, NP_1, DB, NP_1, T2, NDB1, pk(A_1), pk(NP_1), QApk(NP_1))$;
5. Notary NP_1 receives the message.	$recv_3(DB, NP_1, DB, NP_1, T2, NDB1, pk(A_1), pk(NP_1), QApk(NP_1))$;
6. Notary NP_1 performs computations.	match$(j, H(NP_1))$;
7. Notary NP_1 sends a response message to DB.	$send_4(NP_1, DB, NP_1, DB, T3, NDB1, j, QBpk(DB))$;
8. DB receives the response message.	$recv_4(NP_1, DB, NP_1, DB, T3, NDB1, j, QBpk(DB))$;
9. DB performs computations.	match(p,H(A_1)); match(m1,H($NP_1, TIDa, DB, Z, y, T2, NDB2, NA1, pk(DB), p$));
10. DB sends a response message to A_1	$send_5(DB, A_1, NP_1, TIDa, DB, Z, y, T4, NDB2, NA1, pk(DB), p, QDB, m1pk(A_1))$;
11. A_1 receives the response message.	$recv_5(DB, A_1, NP_1, TIDa, DB, Z, y, T4, NDB2, NA1, pk(DB), p, QDB, m1pk(A_1))$