CSP-based verification of TTCB services and GKA TH

Short review of model-checking effort

William Simmonds, QinetiQ

Newcastle, Tuesday 18th February 2003

QinetiQ

WP6 Goals

- Validate results of dependable middleware
  - Formal specification in CSP and verification with FDR
- Provide rigorous general system model which allows for
  - Rigorous definitions of basic concepts
    - Modular systems, synchrony, topology, fault- and attack-models
  - Modular security proofs (composition theorem)
    Combined use of
    - Complexity-theoretic cryptographic proofs
    - Formal methods
    - Best of both research directions

Contents

1. Modelling selected TTCB services in CSP/FDR (D22)
2. Modelling the Group Key Agreement Trusted Host in CSP/FDR (D22)
3. Two-page review of model-checking effort (D4, D7 and D22)

CSP-based verification of TTCB services and GKA TH

CSP-based TTCB Verification

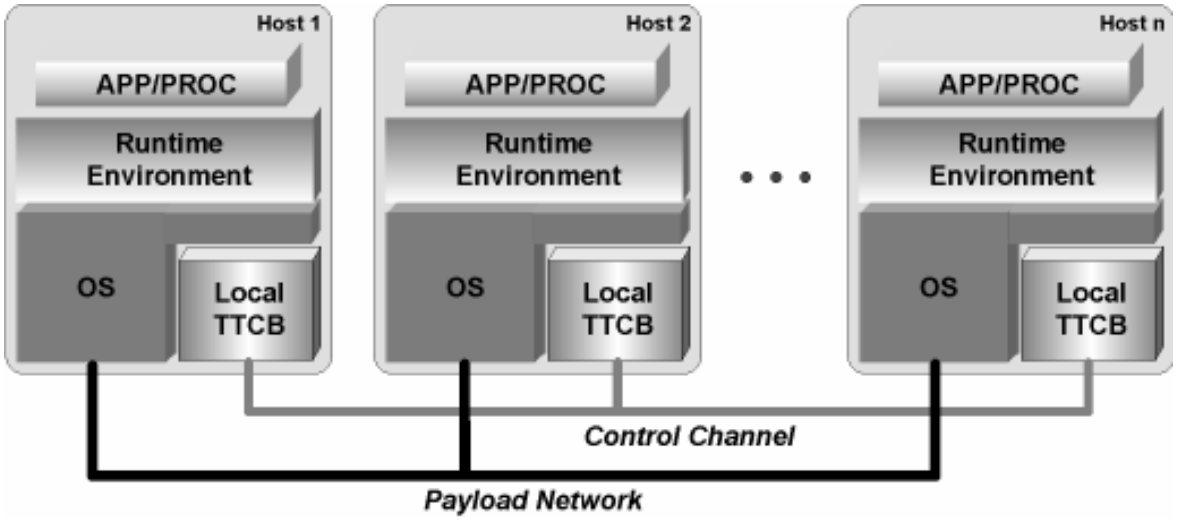
- The verification of TTCB services is important to WP6 because:
  - TTCB is a core MAFTA component
  - It was our rational and basis for research into modelling in the third and final MAFTIA synchrony model, i.e. *hybrid synchrony*
- The two TTCB services we considered were:
  - The *Local Authentication Service* (an example of a *local service*)
  - The *Trusted Block Agreement Service* (an example of a *distributed service*)

Literature

- Our main source of literature is Lisboa abstract:

  The Design of a COTS Real-Time Distributed Security Kernel [1]

TTCB Architecture



Channels

- *Control channel* between the local TTCBs enjoys a certain well-defined degree of reliability and synchrony (AN1-AN8 of [1]), e.g:
  - AN3 states that no more than *Od* omissions may occur in a given interval of time
  - AN4 states that any packet is subject to a maximum (known) delay, T_{send}
- On the other hand, we do not generally make any assumptions about the *payload channel*

Re-Use of Modelling Techniques

- We re-use the notation, conventions and modelling techniques that we've previously deployed wherever possible
- Same channel names and signatures, e.g. *input*, *output*, *send*
- Same common functions
- Same basic modelling approach, in the general case:
  - Separate processes corresponding to distributed nodes running one or more MAFTIA services
  - Depending on synchrony model and link attributes, at most one integrated attacker + network process in parallel with the distributed node processes

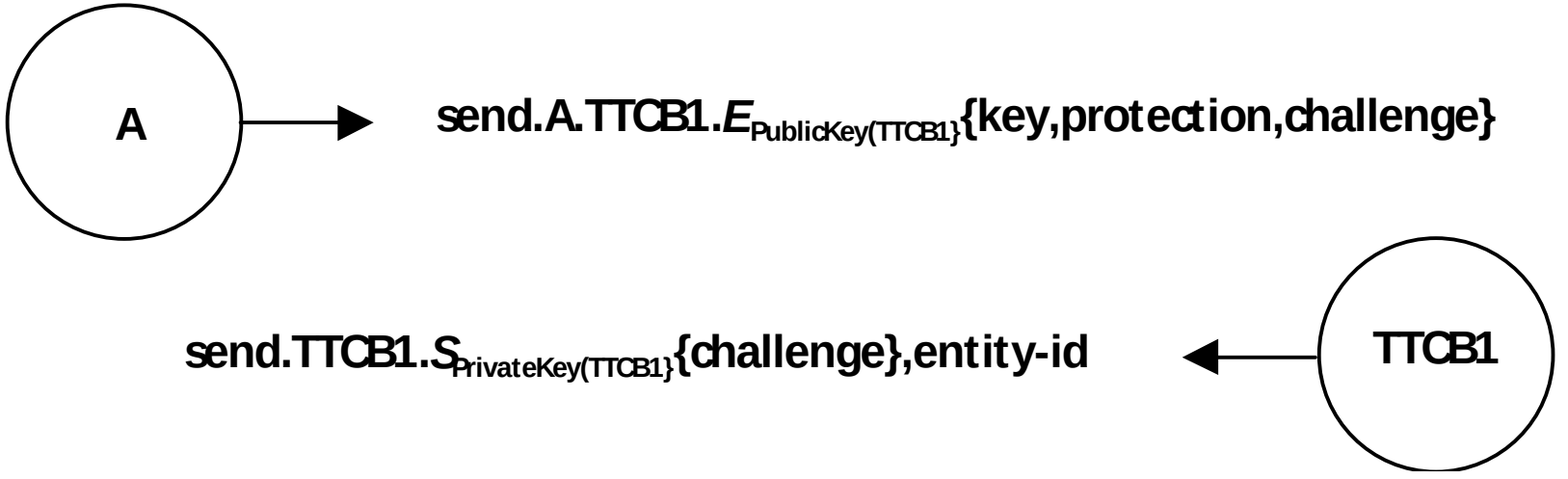
Verifying the Local Authentication Service

The TTCB Local Authentication Service

- The Local Authentication Service allows an entity (node) to agree a secret key with a local TTCB, and thereby establish a secure and authenticated channel between itself and the TTCB
- Our main source of literature is [1]

The protocol

- Protocol is *ostensibly* quite straightforward, requiring only two message-sends (one function call):



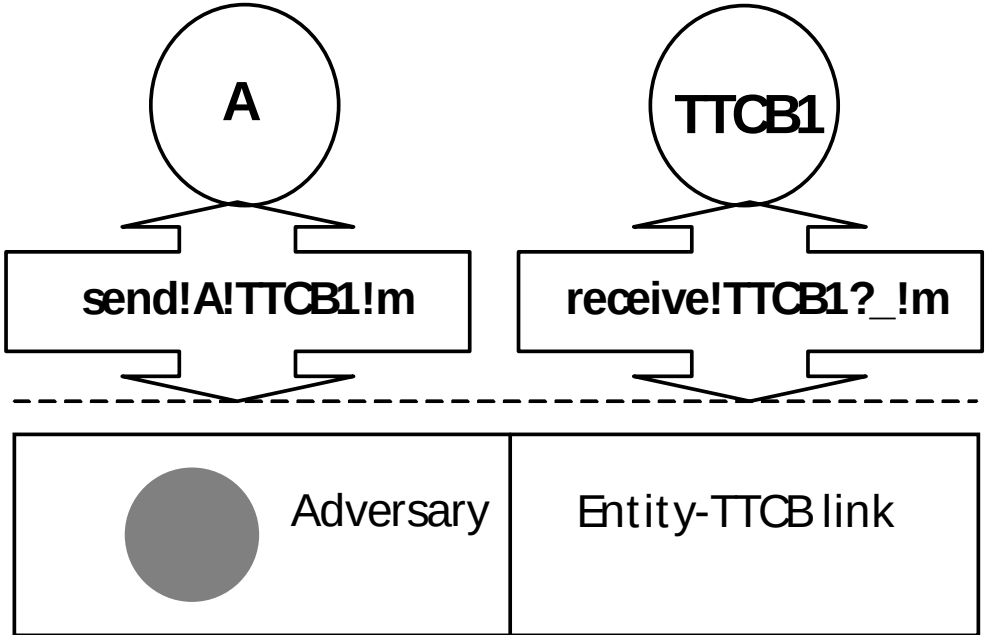
Assumptions

- We made no assumptions about the local link between the entity and its local TTCB re. Reliability, Security, and (not surprisingly) Authentication
- Messages are strongly-typed.
- Usual Dolev-Yao assumptions:
  - Signatures are unforgable
  - Encryption is perfect
- The keys and challenges generated by uncorrupted entities cannot be guessed by the Adversary:
  - This is explicitly assumed and discussed in [1]

Assumptions and Design Decisions

- That the messages are strongly-typed and that the key/challenge pairs cannot be guessed by an Adversary, meant that we could assume that all legitimate messages pertaining to any two runs of the protocol are disjoint
- In addition, all the the four properties required of the service (SK1-SK4 of [1]) cite a single entity and local TTCB
  - N.B. no liveness properties as such (as links are unreliable)
- Thus we could argue that a two node model (one entity + one TTCB) was sufficient to verify the properties
- Our CSP process topology is of the two nodes communicating via an integrated Adversary process + entity-local TTCB link

CSP Process Topology



A note on the importance of dialog

- The generation of the keys, challenges and entity identifiers is very important to the assuredness of the service
- We had to be sure as to who generates what, and be explicit about uniqueness & randomness assumptions:
  - E.g. what happens if an entity were to re-use a key with a different challenge?
- All assumptions were discussed with the protocol authors in a productive dialog that ensued during the course of the verification
  - This was an important lesson learnt from our first verification attempts

Results of LAS modelling

- The Entity ID returned by the local TTCB was not originally encrypted, this could lead to a confused situation in the case that communications between the entity and its local TTCB were insecure
  - The spy could replace entity ID with any value of its choosing
- Hence, it was agreed with the protocol authors that it was better if entity ID was protected
- No attacks found

Verifying the Trusted Block Agreement Service

The Trusted Block Agreement (TBA) Service

- The Trusted Block Agreement Service allows entities to reach consensus in “a secure and timely fashion” via the local TTCBs
- Main source of literature is again [1]

The Four Activities

- The entities may *propose* their decision value to their local TTCB before a notional agreement *start-time*, after which proposals are declined
- The local TTCBs *broadcast* the proposals to each other (requiring *Od*+1 rounds to ensure that all local TTCBs have same data with which to *decide*)
- The local TTCBs wait to *receive* the broadcasts
- Once all proposals are in (or fixed duration after the *start-time*), the local TTCBs *decide* (hopefully for the same value)

The main issues

- Hybrid Synchrony
  - Asynchronous, insecure payload channels between entities and their local TTCBs
  - A control channel between local TTCBs with well-defined characteristics
- A ‘complete’ verification would need to be demonstrably independent in three (four) parameters, i.e.:
  - # of entities
  - # of local TTCBs
  - # of rounds needed in the broadcast stage (= *Od*+1)
  - The *decision function* parameter

Design Designs (1)

- A verification that was independent of the number of entities and local TTCBs would have been very difficult to achieve (although it was discussed)
- Instead, we played safe and verified the TBA for a fixed number of entities (up to 3) and local TTCBs (up to 3)
  - Another lesson learnt from earlier verification attempts

Design Decisions (2)

- The Entity-local TTCB links were assumed to be protected (Authenticated, Integral), and as such the TBA service properties AS1-AS5 were re-interpreted with respect to the local TTCB interface
  - Even the *timeliness* property AS5 was specified w.r.t the local TTCB interface
- The timeliness and reliability of the control channel was abstracted following the design of the TBA protocol
  - Each local TTCB is assumed to broadcast *Od+1* times to ensure delivery, and, when receiving, filter out the duplicates
- The decision function was abstracted away - each service property may be re-interpreted in terms of ‘agreement over sets of *proposals’*

Results

- No attacks found
- We were able to investigate the effects of altering the *timeliness* property AS5
  - *Tagreement* (propagation delay) as it stands is perfectly sufficient (and may even be improved upon - but this remains to be explored)
- During development, compressions over the events shared between the local TTCB processes, thereby greatly reducing the state space

Modelling of the Group Key Agreement Trusted Host

Modelling the Group Key Agreement Trusted Host

- The Group Key Agreement (GKA) Trusted Host (TH) is defined in D22
- The definitions and proofs are phrased in the model of secure reactive systems with adaptive corruption, as defined in D8

What is the Trusted Host?

- The GKA TH is a state transition machine that specifies the ‘desirable’ GKA service.
- The TH is itself a specification - albeit a highly complex one
- It incorporates ‘tolerable imperfections’ that, in practice, one could do little to overcome, for example:
  - The adversary may learn that honest users are preparing to establish a group key by overhearing leaked initial interactions between the machines
- ‘Real’ GKA systems are compared to the TH using the *simulatability* theory

Why model the GKA TH?

- One of the aims of WP6 was to ‘bridge the gap’ between the formal ‘crypto world’ (as exemplified by the rigorous secure reactive systems theory) and the world of automated verification (CSP/FDR)
- Q: Might it be possible to prove some ‘book-keeping’ aspects of simulatability relationships automatically
- If so, that would be very good:
  - Proving simulatability relationships by hand requires much human effort (inevitably open to error)
- A: Yes, using theorem provers (Backes, Jacobi, Pfitzmann)
- A: With model-checkers…? (Gives us debug info, automatic proof, fault injection facility)

A ‘feasibility study’

- We decided to conduct a modest ‘feasibility study’ with the aim of showing the viability of modelling and checking Trusted Host machines in CSP/FDR
- Our example was to be the GKA TH
- Completing a compile-able model was our priority
- Foremost, it would turn out to be an exercise in optimisation
  - It would involve transcribing a complex state-machine into a CSP model that was highly optimised w.r.t. state-space

GKA TH Parameters

- The GKA TH is parameterised by:
  - The number of participating users, *n*
  - The set of a-priori uncorrupted participants, *H*
  - The bound on key length k
    - A key length of k provides 2^k keys
  - The set of session identifiers, *SID*
  - The corruption model, i.e. *static* or *adaptive*

#variables and #configurations

- From these parameters, we calculated the number of variables of the GKA TH, and an *upper bound* on the number of variable states - the configurations - that the machine can be in
- It would be nice to model the TH GKA for at least three users (n=3) and four keys (a key length, k, of 2)

A few calculations later...

#users, n	*Keylength, k*	*|SID|*	#variables	#configurations
1	1	1	7	1296
1	2	1	7	3600
1	1	2	13	6718464
1	2	2	13	51840000
2	1	1	28	7.222041e14
2	2	1	28	4.299817e16
2	1	2	52	1.335242e32
2	2	2	52	4.733037e35
3	1	1	81	?

Reach-ability

- At first sight, the table suggests that it will only be feasible to model the TH for only very small, naïve values of *n*, *k* and *SID*
- However, that could be overly pessimistic, bearing in mind that many of the variable states may not actually be reachable in practice
- This is a quite common modelling phenomenon
- In CSP, it is addressed by writing models so that the bulk of the automated work is moved from FDR's compilation engine to the more efficient state-exploration phase
  - The latter is not so adversely effected by non-reachable states
- How? As much as possible, split process up into many small processes in shared parallel (nearer to interleaving the better)

High-level composition of model

- Our model was composed thus:

((TH_H [|…|] KEYR)

[|{…get_,set_|}|]

VARIABLES

) \ {|…get_,set_|}

  - TH_H = the Trusted Host transitions
  - KEYR = ‘random’ key generation
  - VARIABLES = a variable ‘state keeping’ process

Separation of concerns

- The VARIABLES process is itself composed of many VARIABLE_X processes in parallel, one for each variable of the TH.
- Each VARIABLE_X process is responsible for maintaining the state of a single variable.
- The TH_H process reads and writes to the variables via *get_* and *set_* channels shared with the VARIABLES process
  - TH_H itself has no state
- By splitting the system up like this, we are free to optimise VARIABLES without obfuscating the TH_H process
  - helps tremendously in readability

Example: ‘Initialisation transition

CSP	Original transition
	transition in u ? (init)
get_.state_?u!u!undef ->	**enabled if:** state u,u = undef;
in.u.init ->	
-- state.u.u <- wait ->	
set_.state_.u.u.wait ->	state u,u <- wait;
-- output:	
out_sim!u!init ->	**output:** out sim,u ! (init)
TH_H	

Existential and Universal Quantification

- Matters were complicated by the fact that the TH transitions are dependent on existential and universal quantification over the variables
  - Problematic, as the VARIABLE_X processes do not share state (originally interleaved)
- Our solution involved the VARIABLE_X processes independently synchronising with special ‘control processes’ over *satisfied* channels
- The *satisfied* channels declare whether or not the *exists* or *forall* predicates were satisfied by the individual variables
- Results ‘collected together’ by the ‘control processes’

Use of chase compression

- As it did not matter in what order the *satisfied* events occurred - only that each has a chance to do so - so we could apply *chase* compression to the *satisfied* events
  - This fixed the order in which the *satisfied* events occurred, thereby substantially reducing the number of transitions in the LTS

‘Feasibility Study’ Results

- We just about got a compile-able Trusted Host model - but only for up to two users (our target was at least three).
- Problem is not specifically one of theory
  - No reason why TH and ‘real world’ Sys configurations cannot naively be transcribed in CSP or similar calculi
- Rather, the problem is that the resulting models will invariably be intractably large (in terms of state-space) for FDR to cope with
- But several avenues yet to explore
  - Notably use of ‘Watch Dog’ transforms (such as developed in the Framework 5 DSOS project)

Review of model-checking effort

Cons

- Only partial success in verifying ABBA (and similar probabilistic protocols) data independently of the number of nodes
  - We did get a general ‘DI threshold counting mechanism’ did arise out of our attempts
- Our study implies that the model-checking technology that we used is not currently able to deal with the complexity of the MAFTIA Trusted Host machines
  - This may or may not change with future developments

Pros

- We demonstrated that one can faithfully model protocols in CSP against the three formally defined synchrony models
- Successful work on the verification of liveness properties over networks with ‘eventual delivery’
- We can claim some successful MAFTIA service verifications
  - contract-signing and TTCB services notably

Second WP6 presentation

- Tomorrow there will a presentation on the other strand of WP6's work:
  - Secure reactive systems theory

Questions?