A Collaborative Approach to Autonomic Security Protocols

Hongbin Zhou
Department of Computer Science
University College, Cork
Cork, Ireland
zhou@cs.ucc.ie

Simon N. Foley
Department of Computer Science
University College, Cork
Cork, Ireland
s.foley@cs.ucc.ie

ABSTRACT

This paper considers a new security protocol paradigm whereby principals negotiate and on-the-fly generate security protocols according to their needs. When principals wish to interact then, rather than offering each other a fixed menu of 'known' protocols, they negotiate and, possibly with the collaboration of other principles, synthesise a new protocol that is tailored specifically to their current security environment and requirements. This approach provides a basis for autonomic security protocols. Such protocols are self-configuring since only principal assumptions and protocol goals need to be a-priori configured. The approach has the potential to survive security compromises that can be modelled as changes in the beliefs of the principals. A compromise of a key or a change in the trust relationships between principals can result in a principal self-healing and synthesising a new protocol to survive the event.

1. INTRODUCTION

Networked services and applications are typically commissioned with a fixed repertoire of security protocols that provide for necessary authentication, key-exchange, delegation, non-repudiation, and so forth. Applications and services are expected to negotiate with each other and agree on appropriate security protocols that both can (and are willing to) use. A simple example is a web-service that requires clients to establish SSL based connections. When negotiating a connection, the client and server agree on the version of the protocol to use. While protocols may be designed to support a range of different underlying authentication protocols, and so forth, it is not feasible to expect principals to be, a priori, conversant in all possible protocols. Protocol agnostic approaches such as Jini [12] allow resource providers to register the protocol, that its clients should use, with a Jini Server. However, the Jini server does not have any provision for establishing trust: the communicating parties cannot gain confidence that they have a trustworthy protocol implementation or that they are communicating with the correct party. While more flexible, the provider's protocol is fixed and is not generally suitable for security protocols.

We are investigating the use of protocol synthesis techniques [7, 11, 15, 19] to allow principals negotiate and on-the-fly generate security protocols. When principals wish to interact then, rather than offering each other a fixed menu of 'known' protocols, the protocol negotiation process generates a new protocol that is tailored specifically to their current security environment and requirements. A principal's security environment reflects the keys that it knows, the trust relationships with other principals and any other assumptions it holds.

We conjecture that the applications which use such *Self-configuring* protocols would not require configuration to provide a fixed number of 'known' protocols. Instead, the application designer specifies the necessary security requirements for valid interaction. For example, a server may be willing to accept any connection from an authenticated principal; the security requirements and current environment of the server (and client) are used to synthesis a protocol that meets these goals.

Protocols could be generated on the basis of the security environment of the principals. A change in the security environment of a principal may result in the re-negotiation of a new security protocol. This provides a basis for *survivable* security protocols that have the potential to, in effect, *self-heal* and adapt to recover from changes in the security environment.

These characteristics — self-configuring, self-healing and survivability — are properties that form part of the autonomic computing manifesto [3]. In [9], Foley and Zhou suggest that automated protocol synthesis techniques may provide a basis for autonomic security protocols. In this paper we explore this new paradigm in more depth and consider some of the issues that arise. We propose a logical framework that extends [19] and supports on-the-fly generation of protocols. We extend the basis of [9] with the new paradigm of protocol synthesis by collaboration, whereby the generation of a new protocol becomes a collaborative effort between many principals, each with a potential stake in the outcome.

The paper is organised as follows. Section 2 describes the belief logic on which the framework is based. This logic extends Zhou and Foley's [19] — a BAN-like logic based on the BSW logic [6]. The proposed logic is tailored to support collaborative protocol synthesis, and Section 3 outlines existing research in the area of automatic protocol generation. Section 4 describes how the logic may be used in the collaborative synthesis of a security protocol. A number of

NSPW'04, September 20-23, 2004, Nova Scotia, Canada.

examples are given in Section 5 and in the appendix.

2. THE EXTENDED BSW LOGIC

The BSW Logic [6] is a BAN-style belief logic that uses abstract channels similar to the Spi Calculus [1] to represent keyed communication between principals. The set of principals that can receive and can send messages via a channel C is denoted by its reader set $s(r(C))$ and its writer set $s(w(C))$, respectively. For example, $s(r(C)) = \Omega$ and $s(w(C)) = \{P\}$ represent an authentic channel, whereby any principal (from Ω) can authenticate messages signed by the private key of principal P. $r(C)$ and $w(C)$ represent the principal's properties.

The logic uses the following basic formulae. P,Q range over principals; C represents the channel; X represents a message which can be data or formulae or both; ϕ represents a formula.

- $P \triangleleft X$: Principal P sees message X. Someone has sent X via a channel that P can read.
- $P \triangleleft C(X)$[1]: P sees $C(X)$. Someone has sent a message X via channel C. If P can not read C then P can not discover the content of X.
- $P \mid\sim X$: P once said X. P sent a message contained X at some point in the past. We do not know exactly when the message was sent.
- $P \mid\!\sim X$: P says X. P sent X in the current run of the protocol.
- $\sharp(X)$: Message X is fresh. X has never been said before the current run of the protocol. This is usually true for messages containing nonces.
- $P \models \phi$: P believes that ϕ is true. It does not mean that ϕ is really true, but P believes it.

In the original BSW logic [6], it is not possible to represent the holding of nonces or keys by a principal. A principal can only say what he sees or what he believes, but in the BSW logic a principal cannot generate a message from his knowledge. The following formula is added to represent the holding of a message (in the sense of GNY [10]). The addition of this formula was necessary in order to reason about new beliefs held as a consequence of collaboration.

- $P \ni X$: P holds X. For example, $P \ni r(C)$ represents P holds the property $r(C)$, he can receive message from channel C.

The BSW logic also uses the conventional logic operators $\wedge$ (conjunction), $\vee$ (disjunction) and $\rightarrow$ (implication) from propositional logic and some basic notation from set theory.

In the BAN logic principals are treated as trustworthy and in GNY there are fixed axioms for reasoning about the trustworthiness of a principal. In BSW, the rules about the trustworthiness of a principal are expressed as formulae as part of the assumptions of the protocol.

[1]Note that we do not use the related formula $P \triangleleft X \mid C$ from [6] which defines that P sees message X via channel C since it can be replaced by the other formulae in the deduction axioms without any loss of expressiveness.

EXAMPLE 1. *(Adapted from [6]). Mutual authentication between principals A and B may be expressed as goals:*

$$G_1 \triangleq A \models (B \mid\!\sim (A, Na))$$
$$G_2 \triangleq B \models (A \mid\!\sim (B, Nb))$$

where Na is a nonce (and assumptions include $A \models \sharp(Na)$, and so forth). We assume that A and B share symmetric keys (abstracted as channels Cas and Cbs, respectively) with third party S. These assumptions are defined as follows.

$$S \ni r(Cas)); A \ni r(Cas);$$

$$A \models (s(w(Cas)) = \{A, S\}); S \models (s(w(Cas)) = \{A, S\});$$

A further assumption is that A trusts S as a trusted third party:

$$A \models ((S \mid\!\sim \phi_1) \rightarrow (S \models \phi_1))$$
$$A \models ((S \models (B \mid\sim \phi_2)) \rightarrow (B \mid\sim \phi_2))$$

for arbitrary ϕ_1, ϕ_2. These formulae reflect A's belief that S is honest and that S is competent in deciding whether B at some time in the past said some message. B has similar beliefs. $\triangle$

In addition to basic axioms about sets, the original BSW logic uses five core axioms.

S1 Seeing. If P receives a message X via a channel C, and P can read this channel, then P can see the message.

$$\frac{P \triangleleft C(X), P \ni r(C)}{P \triangleleft X}$$

F1 Freshness. If P believes another principal Q once said a message X and P believes that X is fresh, then P believes that Q says X.

$$\frac{P \models (Q \mid\sim X), P \models \sharp(X)}{P \models (Q \mid\!\sim X)}$$

F2 Freshness. If P believes X is fresh, then P believes X' which contains X is also fresh.

$$\frac{P \models \sharp(X)}{P \models \sharp(X)'}$$

I1 Interpretation. If P believes he receives a message via C, then he believes that the message was said by someone, who he believes is able to write channel C except himself.

$$\frac{P \triangleleft C(X), P \ni r(C), P \models (s(w(C)) = \mathcal{W})}{P \models \bigvee_{\forall Q_i \in \mathcal{W} \setminus \{P\}} (Q_i \mid\sim X))}$$

R1 Rationality. This is the well-known K axiom of modal logic: if P believes ϕ_1 implies ϕ_2, and believes that ϕ_1 is true, then he believes that ϕ_2 is true.

$$\frac{P \models (\phi_1 \rightarrow \phi_2), P \models \phi_1}{P \models \phi_2}$$

For the purposes of this paper we propose three further axioms for use when reasoning about message holding.

H1 Holding. If P holds X and Y, and P knows how to apply function F to them, then P can holds $F(X, Y)$.

$$\frac{P \ni X, P \ni Y}{P \ni F(X, Y)}$$

S2 Seeing. If P sees a message X, then P holds X.

$$\frac{P \triangleleft X}{P \ni X}$$

B1 Self-Believing. If P believes that P believes ϕ, then P believes ϕ. It means P can tell what he believes.

$$\frac{P \models (P \models \phi)}{P \models \phi}$$

3. SECURITY PROTOCOL SYNTHESIS

Security Protocols are widely used in distributed systems for authentication, key exchange and other security requirements. Designing well behaved security protocols is a challenging task since protocols often contain subtle flaws that are difficult to find. In the last 20 years, many approaches for verifying properties of security protocols have been developed such as [5, 10, 8, **?**, 16, 14, 18]. However, little work has been carried out on systematic approaches to the design and development of security protocols

Abadi and Needham [2] set out ten principles that help designers avoid classes of known protocol flaws. However, the principles are neither necessary nor sufficient: designers can not necessarily design new protocols by obeying only these principles. A number of formal design approaches for security protocols have been proposed. Alves-Foss and Soule [4] describes a weakest-precondition based approach for the design of security protocols. The Simple/BSW logic [6] is a BAN-like logic that provides synthesis rules to guide the protocol designer in the manual systematic calculation of a protocol from its goals.

We are interested in the automatic generation of a protocol from its protocol goals and assumptions. Existing research includes Clark and Jacob's evolutionary search [7], Perrig and Song's Automatic Protocol Generator (APG) [15] and Zhou and Foley's Automatic Security Protocol Builder (ASPB) [19]. All approaches automatically search a large space of candidate protocols that is far larger than could be considered via a manual design.

Starting from a set of assumptions, [7] uses the original inference rules of the BAN logic to systematically test whether candidate protocols uphold the protocol goals. The evolutionary approach starts from a set of assumptions, and uses a fitness function to guide the application of BAN rules in a forward manner until a valid protocol is arrived at. However, providing an accurate fitness function for a protocol is very difficult, and in cases potentially impossible [7]. Perrig and Song's APG [15] uses heuristics to select random candidate protocols that are in turn checked for validity using the Athena [17] security protocol checker.

In [19] Zhou and Foley propose the Automatic Synthesis Protocol Builder (ASPB) which combines and automates the manual synthesis rules from the BSW Logic together with Guttman's manual design process [11]. The synthesis rules of the BSW logic are adapted to guide an automatic backwards search for a sub-protocol from a single goal. Given a number of individual goals, an automated technique is proposed to combine synthesised sub-protocols into final candidate protocols. For the purposes of the research described in this paper, ASPB has been extended to support the Extended BSW logic in order to support collaborative synthesis.

4. AUTONOMIC SECURITY PROTOCOLS

In [9] a basic protocol synthesis protocol (BPSP) is outlined. A protocol initiator I requests connection to R. R responds by passing details of its protocol goal G_R and the assumptions A_R that it currently holds. Principal I uses its own assumptions and those presented by R to synthesise a new protocol P that meets their respective goals. This protocol is returned to R, which attempts to validate using the validation tool. If validation is successful then I and R install and engage in the protocol P.

In this paper we generalise upon this scheme such that a principal may draw on the help of other principals during protocol synthesis. This collaboration is necessary when the principal does not hold sufficient beliefs to synthesise the *valid* protocol on its own. By a *valid* protocol we mean that the protocol meets the specified goals. The protocol is valid in our logic. However if the user needs to make sure it is a real secure protocol, it has to be validated by other stronger security protocol checkers. It is a way to describe the properties of the protocol to the user in a comprehensible way.

Figure 1 depicts the agents and their exchanges in this collaborative protocol synthesis protocol (CPSP). A principal receives a protocol goal specification GS from which it is requested to synthesise a suitable protocol PS. This is coordinated by the Logic Deriver. Synthesis is carried out based on the (long-term and short term) assumptions held by the principal. A principal's assumptions are represented in its long-term and short-term assumption databases. Long-term assumptions represent a principal's basic beliefs about others, channels that represent long-term keys, and so forth. Short-term assumption are short-term assumptions that assist in the synthesis of a particular protocol and include assumptions regarding nonces, short-term keys and temporary beliefs.

The desired goal GS is synthesised using ASPB into a collection of partial protocol steps and further protocol subgoals (which are synthesised further). When a sub-goal cannot be synthesised further, assistance is sought from trusted others; the subgoal (QS) is forwarded to the Helper agent of another principal; which, in turn, attempts to synthesise the protocol as described above, generating the protocol AS. These principals may use further principals to help them synthesise their own subgoals. To what extent a principal trusts a potential collaborator is specified as part of the assumptions (beliefs) of the principal.

4.1 Protocol Agent Interactions

All beliefs that are held by principals are represented within the (Extended) BSW Logic. These include the (long-term and short-term) assumptions held by principals and the goals and protocol steps that are exchanged between their (Deriver and Helper) agents.

Appendix A gives as an example of a series of exchanges QS, AS of beliefs between two principals during the synthesis of a goal GS. The goal specification GS defines the goals to be synthesised (by the Deriver agent). Normally, the assumption section is empty, however it can be used for short-term assumptions used in the establishment of the protocol (for example, regarding the freshness of a nonce).

The query specification QS is used to request help in the synthesis of a subgoal that is included in in the goal section of QS. The assumption section may contain the Deriver's

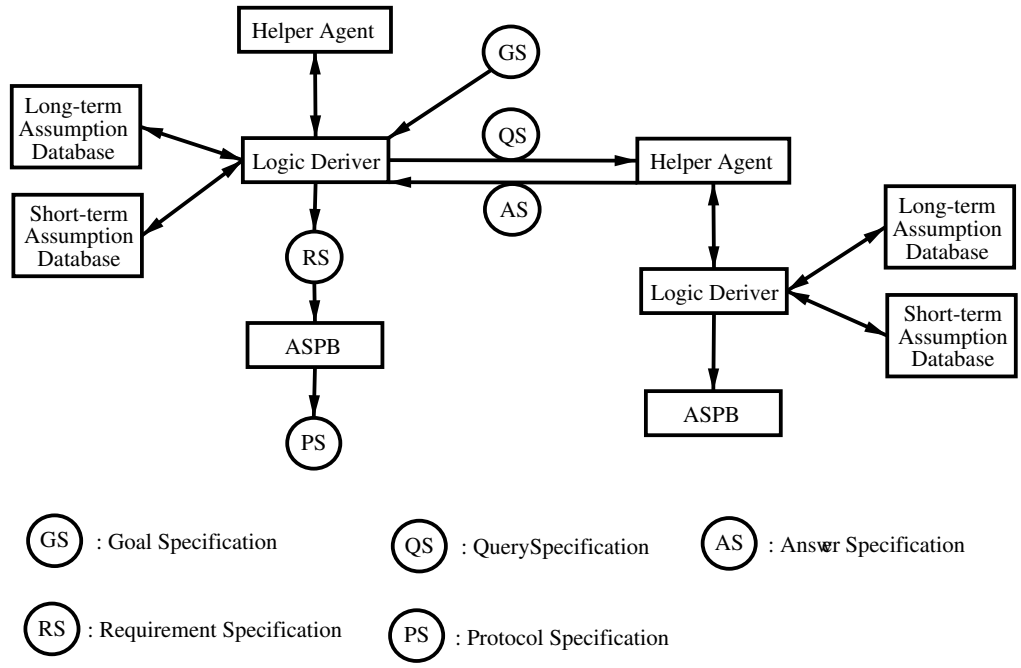

Figure 1: Collaborative Protocol Synthesis Protocol

useful assumptions for this subgoal as an optional section. The requester may sends its QS to a selected principal who is trusted by it and may help it, or a group of principals. It depends on the requester's asking strategy.

On receiving QS, a Helper agent attempts to synthesise the subgoal. The answer specification AS provides assumptions held by the helper (unknown to the Deriver) and that were used by the Helper agent during the synthesis of the subgoal. If the Deriver agent trusts the Helper agent (that is, $Deriver \models (Helper \models \phi \rightarrow \phi)$) then these Helper assumptions[2] can be believed by the Deriver and be added to its assumption database. The protocol to be synthesised by ASPB is derived from the new (and existing) assumptions as specified in the requirement specification RS.

From the requirement specification, ASPB may generate different protocols. In the current implementation, ASPB choose one of them randomly: all generated protocols meet the principals' goals. By adding cost functions to protocol operations, the problem becomes one of searching, for an optimal protocol, an ordered set of generated protocols. This is a topic for future work.

4.2 Extended BSW Protocol Synthesis

The BSW Logic includes a synthesis technique that can be used to guide the systematic calculation of a protocol from its goals. Synthesis rules take the general form

$$\begin{array}{ll} G & \\ \hookrightarrow & G_1 \\ \hookrightarrow & ... \end{array}$$

which means that in order to reach the goal G, all subgoals G_1, G_2, ... have to be reached. A goal G can have the form G'/G'', which means that either G' or G'' have to be reached. In [19] we adapted the synthesis rules in [6] to form a a series of heuristic rules used to guide the automatic backward search for candidate protocols from their goals. These heuristics are effectively theorems within the logic and, therefore, any protocol synthesised under these rules is valid (within the constraints of the logic). The synthesis heuristics, adapted to support the Extended BSW Logic proposed in Section 2 are defined as follows.

Heur1 To hold $F(X,Y)$, P must hold X and Y.

$$\begin{array}{ll} P \ni F(X,Y) & \\ \hookrightarrow & P \ni X \\ \hookrightarrow & P \ni Y \end{array}$$

Heur2 To hold X, P must see the message X.

$$\begin{array}{ll} P \ni X & \\ \hookrightarrow & P \triangleleft X \end{array}$$

Heur3 To see message X, P must receive $C(X)$ and be able to read C.

$$\begin{array}{ll} P \triangleleft X & \\ \hookrightarrow & P \triangleleft C(X) \\ \hookrightarrow & P \ni r(C) \end{array}$$

[2]Note that it will not detect contradictory assumptions held by two parties or a single party, since the derivers search for the assumptions that they require, and there should no contradictory assumptions in the same searching tree within the logic. Therefore, we do not consider this problem.

Heur4 To believes ϕ, a principal must have self-belief.

$$P \models \phi \quad \hookrightarrow \quad P \models (P \models \phi)$$

Heur5 To believe Q says X, P must believe that Q said X and X is fresh.

$$P \models Q \mid\!\sim X \quad \hookrightarrow \quad P \models Q \mid\sim X \quad \hookrightarrow \quad P \models \sharp(X)$$

Heur6 To believe that message X is fresh, P must believe that some part X' of X is fresh.

$$P \models \sharp(X) \quad \hookrightarrow \quad P \models \sharp(X')$$

Heur7 To believe that Q said X, P has to receive X via a channel C that he can read and that he believes it can be written only by Q, or P and Q. Furthermore, Q has to see X.

$$\begin{aligned} P \models Q \mid\sim X \quad & \hookrightarrow \quad P \triangleleft C(X) \\ & \hookrightarrow \quad P \ni r(C) \\ & \hookrightarrow \quad P \models (s(w(C)) = \{Q\})/ \\ & \qquad\quad P \models (s(w(C)) = \{P, Q\}) \\ & \hookrightarrow \quad Q \triangleleft X \end{aligned}$$

If X is a formula and P believes that Q is honest, then Q must also believe X.

$$\begin{aligned} P \models Q \mid\sim X \quad & \hookrightarrow \quad P \triangleleft C(X) \\ & \hookrightarrow \quad P \ni r(C) \\ & \hookrightarrow \quad P \models (s(w(C)) = \{Q\})/ \\ & \qquad\quad P \models (s(w(C)) = \{P, Q\}) \\ & \hookrightarrow \quad Q \models X \\ & \hookrightarrow \quad P \models ((Q \mid\!\sim X) \rightarrow (Q \models X)) \end{aligned}$$

Heur8 To believe ϕ_1, P must believe ϕ_2 and $\phi_2 \rightarrow \phi_1$.

$$P \models \phi_1 \quad \hookrightarrow \quad P \models \phi_2 \quad \hookrightarrow \quad P \models (\phi_2 \rightarrow \phi_1)$$

4.3 The Logic Deriver

The Automatic Prototol Sythesis Builder (ASPB) [19] has been adapted to use these new extended heuristic rules to guide its backwards search for candidate protocols [3]. The Logic Deriver directs the ASPB, by building the requirement specification to be synthesised with the collaboration of other Helper agents. The Logic Deriver takes a set of protocol goals G_P (obtained by parsing goal specification GS) and assumptions A (obtained from the principal's assumption databases). and returns a synthesised protocol. Algorithm 1 describes this process.

[3]BAN-style logics are much simpler than programming languages (e.g., no looping structures and thus it is not necessary to search for loop invariants), and, therefore, a few simple production rules guide the search efficiently without external intervention.

```
G = G_P;
while ¬ empty(G) do
  g = choose(G);
  G = G \ g;
  State s = syn(g);
  if empty(s) then
    Spec qspec = genQS(s);
    Spec aspec = helper(qspec);
    s = genState(aspec);
  end if
  G' = goals(s);
  A' = assumptions(s);
  G = add(G, G');
  A = add(A, A');
end while
Spec rspec = genRS(G_p, A);
return rspec;
```

Algorithm 1: Spec *deriver*(Set G_p, Set A)

Operation $choose(G)$ picks an arbitrary goal from the goal set G.

Operation $syn(g)$ applies the currently applicable heuristic rules to the goal g, generates all conclusions by all applicable logical rules, returns a state s containing a set of assumptions and subgoals derived from g. If s is empty, then there are no local reachable assumptions for the goal and, in this case, the Deriver agent generates a query specification $qspec$ using operation $genQS(s)$.

Operation $helper(qspec)$ sends $qspec$ to other principals who are willing to help with this query specification, and waits the answer specification $aspec$ as a synthesising result set. A security protocol between deriver and helper is used to ensure the authenticity of the answer specification. The details will be discussed in next section.

After the application of the heuristic rules, the Deriver agent checks whether the resulting subgoals are interim or terminal subgoals. If the subgoal matches a protocol message or an assumption then it is a terminal subgoal and the searching at this point will complete. Otherwise, in the case of an interim subgoal it added to the current goal set G. If no heuristic rule can be applied to a subgoal then it is a terminal subgoal. If it matches an assumption from local assumption database, or has the format $P \triangleleft C(X)$ (a protocol step), then it is a local reachable terminal subgoal.

4.4 The Helper Agent

The Helper agent accepts query specifications and uses its own local Deriver agent to determine the assumptions that that are needed to synthesise the protocol. This exchange of protocol specifications requires a security protocol in itself; an attacker may attempt to trick one of (or both) of the principals into using a different protocol. Rather than fixing on a fixed protocol, we follow our own autonomic security protocol paradigm and envisage the parties involved synthesising a protocol that will be used to exchange protocol specifications.

Suppose that agents are not concerned with establishing a secure protocol exchange then the query QS and answer AS protocol specifications are exchanged as plain text. For example, suppose that the requester A has subgoal $B \triangleleft (A, Na)$. A doesn't know which channel B can read, and doesn't care about it. What she requires is that B can read from some channel that she can write to. If, during the synthesis of the

protocol, an intruder C masquerading as the Helper agent modifies the subgoal or the returned assumptions leading to then B may not see the message, and B may not send the message. The generated protocol will not run to completion. Though the protocol can be generated in a false way, the intruder C can not masquerade as the intended principal B.

Alternatively, suppose that the requester A needs to believe that B received her query specification QS and sent his answer specification AS back to her recently. The synthesis exchange protocol goal

$$G_3 \triangleq A \models (B \mid\sim (A, Na, AS \rightarrow QS))$$

means that A believes B says AS in the same round of the helping protocol which A sent QS. To believe B can help A with the query specification QS, A should believe that B is honest and competent in deciding what is the answer specification AS for QS. These assumptions may be expressed as:

$$A \models ((B \mid\sim \phi) \rightarrow (B \models \phi))$$
$$A \models (B \models (AS \rightarrow QS) \rightarrow (AS \rightarrow QS))$$

B has a further assumption:

$$B \models (AS \rightarrow QS)$$

which means B believes that AS can be synthesised to QS.

finally, suppose that the helper B also needs to believe that A sent her query specification to him recently. The goal may be expressed as:

$$G_4 \triangleq B \models (A \mid\sim (QS, Nb, B))$$

The helper B may also have three levels of trusted beliefs for A [4] We list them as follows:

- $B \models (B \lhd (Na, A, QS) \rightarrow (Na, A, QS))$: If principal B sees a message contained (Na, A, QS), B believes A sent (Na, A, QS).
- $B \models (A \mid\sim (Na, A, QS) \rightarrow (Na, A, QS))$: If B believes A sent a message contained (Na, A, QS) at some point in the past, B believes A sent (Na, A, QS).
- $B \models (A \mid\sim (Na, A, QS) \rightarrow (Na, A, QS))$: If B believes A says a message contained (Na, A, QS), B believes A sent (Na, A, QS).

The helping protocol is generated by the principal who needs help based on his assumptions. A typical helping process between A and B may have three steps. At first, A generates the help protocol(HP) achieving G_3 and G_4. Then, A generates a pre-help protocol for the goal

$$G_5 \triangleq B \models (A \mid\sim (B, Nb, HP))$$

and sends it to B. B verifies the help protocol. A and B execute the help protocol in the end.

[4]The underlying BSW logic is based on a small number of core axioms. A variety of trust relationships such as principal truthfullness and honesty are specified as assumptions held by principals. We assume that the specifier of these assumptions accurately reflects the desired trust relationships.

5. EXAMPLES

In this section, two examples are used to demonstrate how our paradigm operates. In the first example, a trusted third party(TTP) is used to help principals to synthesise protocols. In the second example, two TTPs are used to help them.

EXAMPLE 2. *WebCom D is in UCC and provides a special campus computing service to all students in Irish universities. A student from an Irish universities who wishes to use WebCom, must prove his student identity to Webcom. However, different universities have different requirements for authentication. Webcom uses a signature key K_D. TCD students use symmetric keys that are shared with their department. For example, student Bob B shares a symmetric key K_{ab} with his department manager Alice A. When B wants to use WebCom D, D has a goal*

$$G_6 \triangleq D \models (B \mid\sim (D, Nd_1))$$

which means D believes when B says (D, Nd), B proves his identity to D.

WebCom D's long-term assumption database is as follows.

$$D \ni w(Cd), D \ni r(Ca),$$
$$D \models (s(w(Ca)) = \{A\}),$$
$$D \models \sharp(Nd_1), D \models \sharp(Nd_2),$$
$$D \models ((A \mid\sim (B \mid\sim X)) \rightarrow (B \mid\sim X)),$$
$$D \models ((A \mid\sim \phi) \rightarrow (A \models \phi)),$$
$$D \models ((A \models (AS \rightarrow QS) \rightarrow (AS \rightarrow QS))$$

D believes A is honest and competent in deciding what is the answer specification and what B said.

Bob B's long-term assumption database is as follows.

$$B \ni r(Cab), B \models (s(w(Cab)) = \{A, B\})$$

Alice A's assumption Database:

$$A \ni r(Cab), A \ni r(Cd),$$
$$A \models \sharp(Na_1), A \models \sharp(Na_2),$$
$$A \models (s(w(Cab)) = \{A, B\}),$$
$$A \models (s(w(Cd)) = \{D\}),$$
$$A \models (AS \rightarrow QS)$$

Other than the symmetric keys shared with their students, Alice can also use signature key K_a to sign messages.

At the epoch, W uses his Logic Deriver to obtain the assumptions necessary for G_6. Having using Heuristic rules 5,8,7, D gets subgoal

$$A \models (B \mid\sim (D, Nd_1))$$

which he can not synthesise. However, D trusts that A can provide some help and launch a round of the help protocol with A. B generate a query specification QS (see appendix A for details). The help protocol achieves the following Goals,

$$D \models (A \mid\sim (D, Nd_2, AS \rightarrow QS)),$$
$$A \models (D \mid\sim (QS, Na_1, A))$$

D obtains the help protocol by using ASPB.

The help protocol HP_1 is

$$D \triangleleft (Na_1, A),$$
$$A \triangleleft Cd(Na_1, A, Nd_2, D, QS),$$
$$D \triangleleft Ca(Nd_2, D, AS \rightarrow QS).$$

A should accept this protocol HP_1, the goal expressed as

$$A \models (D \mid\!\sim (Na_2, A, HP_1))$$

The pre-help protocol for this goal can express as

$$D \triangleleft (A, Na_2),$$
$$A \triangleleft Cd(A, Na_2, HP_1).$$

Having negotiated these protocols, D obtains the required assumptions. D then generates the security protocol P_1 to authenticate B's identity. The protocol P_1 is expressed as

$$B \triangleleft (D, Nd_1),$$
$$A \triangleleft Cab(D, Nd_1),$$
$$D \triangleleft Ca(B \mid\sim (D, Nd_1).$$

△

The next example is similar to Example 2, however, D does not trust A any more. The Department manager Trent T in UCC is trusted by D.

EXAMPLE 3. *In this scenario, D has the same goal G_6. D's new assumption database:*

$$D \ni w(Cd), D \ni r(Ct),$$
$$D \models (s(w(Ct)) = \{T\}),$$
$$D \models \sharp(Nd_1), D \models \sharp(Nd_2),$$
$$D \models ((T \mid\sim \phi) \rightarrow (T \models \phi)),$$
$$D \models ((T \models \phi) \rightarrow \phi)),$$
$$D \models ((T \models (AS \rightarrow QS) \rightarrow (AS \rightarrow QS))$$

D believes T is honest and competent in deciding what is the answer specification.

T's assumption database:

$$T \ni w(Ct), T \ni r(Ca),$$
$$T \models (s(w(Ca)) = \{A\}),$$
$$T \models (s(w(Cd)) = \{D\}),$$
$$T \models \sharp(Nt),$$
$$T \models ((A \mid\sim (B \mid\sim X)) \rightarrow (B \mid\sim X)),$$
$$T \models ((A \mid\sim \phi) \rightarrow (A \models \phi)),$$
$$T \models ((A \models (AS \rightarrow QS) \rightarrow (AS \rightarrow QS))$$

T believes A is honest and competent in deciding what is the answer specification and what B said.

A changes her assumption

$$A \models (s(w(Cd)) = \{D\})$$

to

$$A \models (s(w(Ct)) = \{T\})$$

the other assumptions are the same as in Example 2 and B has the same assumptions as in Example 2.

To synthesise a suitable protocol, D launches a round of the help protocol with T, and T launches another round of help protocol with A. Finally, D generate a protocol P_2 to authenticate B. P_2 is expressed as

$$B \triangleleft (D, Nd_1),$$
$$A \triangleleft Cab(D, Nd_1),$$
$$T \triangleleft Ca(B \mid\sim (D, Nd_1),$$
$$D \triangleleft Ct(B \mid\sim (D, Nd_1).$$

△

From example 3 we can see this new paradigm can generate n-party protocols based on their beliefs. It is not unlike a cascading authentication protocol such as PGP [?] with the the trusted chain generated automatically by the security agents. If two parties have never met before, they may also try to establish security protocols. In this case, one party must exactly know the other's unique identity.

6. CONCLUSIONS

This paper considers a new paradigm whereby principals negotiate and on-the-fly generate suitable security protocols. When principals wish to interact then, rather than offering each other a fixed menu of 'known' protocols, the protocol negotiation process generates a new "session" protocol that is tailored specifically to their current security environment and requirements [5]. We believe that this paradigm shift provides a basis for autonomic security protocols, that is protocols that are self-configuring, survivable and self-healing. Protocols can be generated on the basis of the security environment of the principals and their collaborators.

In [9] we suggest that automated protocol synthesis techniques may provide a basis for autonomic security protocols. In this paper we explore this new paradigm in more depth and also consider some of the issues that arise. In particular, we propose a new paradigm of protocol synthesis by collaboration, whereby a principal may draw upon the help of other principals during protocol synthesis. This collaboration is necessary when the principal does not hold sufficient beliefs to synthesise the protocol on its own. In this paper we outline how the Automatic Synthesis Protocol Builder [19] has been extended to support this paradigm.

There are many interesting research directions in this area. In our current prototype, the Deriver and Helper agents communicate with each other using a fixed exchange protocol. In Section 4.4 we argued that this exchange protocol could be synthesised automatically based on the requirements of the principals involved. We are exploring how techniques such as [13] can be used to translate (generated) protocol specifications into executable code, which in turn would form the helper protocols that are used during the synthesis. We are also exploring how this framework can be used to manage more sophisticated protocols such as non-repudiation and delegation.

7. ACKNOWLEDGMENTS

We are grateful for helpful feedback from the anonymous referees and workshop attendees. This work is supported by the Boole Centre for Research in Informatics, University College Cork under the HEA-PRTLI scheme and by

[5]However, principals must express their goals and assumptions in an appropriate way. Any improper expression may lead to unexpected protocol behaviour.

the Center for Unified Computing under the SFI webcom-G project.

8. REFERENCES

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[2] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. In *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privace*, pages 122–136. IEEE Computer Society Press, 1994.

[3] A.G.Ganek and T.A.Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, January 2003.

[4] J. Alves-Foss and T. Soule. A weakest precondition calculus for analysis of cryptographic protocols. In *DIMACS Protocols Workshop*, 1997.

[5] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[6] L. Buttyán, S. Staamann, and U. Wilhelm. A simple logic for authentication protocol design. In *11th IEEE Computer Security Foundations Workshop*, pages 153–162. IEEE Computer Society Press, 1998.

[7] John A Clark and Jeremy L Jacob. Searching for a solution: Engineering tradeoffs and the evolution of provable secure protocols. In *Proceedings 2000 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2000.

[8] D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[9] S. N. Foley and H. Zhou. Towards an architecture for autonomic security protocols. In *11th International Security Protocols Workshop*, Cambridge, UK, April 2003. www.cs.ucc.ie/~s.foley/pubs/secprot03.ps.

[10] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings of the IEEE 1990 Symposium on Security and Privacy*, pages 234–248, Oakland, California, May 1990. IEEE Computer Society Press.

[11] Joshua D. Guttman. Security protocol design via authentication tests. In *Proceedings of 15th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, April 2002.

[12] Sun Microsystem Inc. Jini technology core platform specification version 1.2. www.jini.org, November 2001.

[13] J. Millen and F. Muller. Cryptographic protocol generation from CAPSL. Technical Report SRI-CSL-01-07, SRI International, December 2001.

[14] L.C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[15] A. Perrig and D.X. Song. Looking for diamonds in the desert: extending automatic protocol generation to three-party authenticatoin and key agreement protocols. In *Proceedings of 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, July 2000.

[16] A.W. Roscoe. Using intensional specifications of security protocols. In *Proceedings of the Computer Security Foundations Workshop*, pages 28–38. IEEE Press, 1996.

[17] D. X. Song. Athena: a new efficient automated checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1999.

[18] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: why is a security protocol correct? In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, 1998.

[19] H. Zhou and S. N. Foley. Fast automatic synthesis of security protocols using backward search. In *ACM Workshop on Formal Methods in Security Engineering (FMSE)*, Washington DC, USA, October 2003.

APPENDIX

A. AUTHENTICATION WITH TTP

A.1 Goal Specification

declarations {
  Principal D, B;
  Nonce Nd_1;
}
assumptions {
  $D \models \sharp(Nd_1)$;
}
goals {
  $D \models (B \mid\!\sim (D, Nd_1))$; /*$G_1$*/
}

A.2 Query Specification

declarations {
  Principal D, B, A;
  Nonce Nd_1;
}
goals {
  $A \models (B \mid\!\sim (D, Nd_1))$; /*$G_2$*/
}

A.3 Answer Specification

declarations {
  Channel Cab,Cp;
  Principal D, B, A;
  Nonce Nd_1;
}
assumptions {
  $A \ni r(Cab)$;
  $A \models (s(w(Cab) = \{A, B\})$;
  $B \ni r(Cp)$;
}
goals {
}

A.4 Requirement Specification

declarations {
  Channel Cab, Cd, Ca, Cp;
  Principal A, B, D;
  Nonce Nd_1;
  Message X;

Formula ϕ;
}
assumptions {
$A \models (s(w(Cab)) = \{A, B\})$;
$D \models (s(w(Ca)) = \{A\})$;
$D \ni r(Ca)$;
$B \ni r(Cab)$; $A \ni r(Cab)$;
$A \ni r(Cp)$; $A \ni w(Cp)$;
$B \ni r(Cp)$; $B \ni w(Cp)$;
$D \ni r(Cp)$; $D \ni w(Cp)$;
$D \models \sharp(Nd_1)$;
$D \models ((A \mid\!\sim \phi) \rightarrow (A \models \phi))$;
$D \models ((A \models (B \mid\!\sim X)) \rightarrow (B \mid\!\sim X))$;
}
goals {
$D \models (B \mid\!\!\sim (D, Nd_1))$; /*$G_1$*/
}

A.5 The protocol Specification

declarations {
Channel Cab, Cd, Ca, Cp;
Principal A, B, D;
Nonce Nd_1;
Message X;
Formula ϕ;
}
assumptions {
$A \models (s(w(Cab)) = \{A, B\})$;
$D \models (s(w(Ca)) = \{A\})$;
$D \ni r(Ca)$;
$B \ni r(Cab)$; $A \ni r(Cab)$;
$A \ni r(Cp)$; $A \ni w(Cp)$;
$B \ni r(Cp)$; $B \ni w(Cp)$;
$D \ni r(Cp)$; $D \ni w(Cp)$;
$D \models \sharp(Nd_1)$;
$D \models ((A \mid\!\sim \phi) \rightarrow (A \models \phi))$;
$D \models ((A \models (B \mid\!\sim X)) \rightarrow (B \mid\!\sim X))$;
}
protocols {
$B \triangleleft (D, Nd_1)$;
$A \triangleleft Cab(D, Nd_1)$;
$D \triangleleft Ca(B \mid\!\sim (D, Nd_1)$;
}
goals {
$D \models (B \mid\!\!\sim (D, Nd_1))$; /*$G_1$*/
}