We’ve been looking at what we call the security protocol negotiation problem. People who use security protocols to authenticate, do key exchange, or whatever, are typically stuck with using whatever protocol is available. If the participants can’t agree on a suitable protocol, then they don’t get to use each other’s services. We are interested in self-configuring security protocols where the participants have to say what their requirements are and synthesise or configure their own suitable protocol that would meet their respective goals. If we have this idea of self-configuring security protocols then, as a consequence, we also have self-healing security protocols. If the requirements change, or if the assumptions about our principals and keys change, then we have this ability to reconfigure and produce a security protocol that will meet those new goals, based on those assumptions.

To motivate the work we use a very simple example based on the movie ‘When Harry Met Sally’. There’s these two people in the movie and they don’t get on at all. The problem is that they both have their own strategies for interacting, for getting what it is they want. And so Harry has a bunch of strategies that he uses when he tries to chat up women, and Sally has her own strategies, and they can’t agree on a common strategy, so there’s no interaction. Only after a lot of time do they realise that they need to reconfigure their strategies, use different ones that will be consistent with each other, and so what we have is self-configuring and self-healing, and survival strategies. And these are properties (among others) of autonomic systems. The properties that we’re interested in are self-configuring, i.e. the ability to compute their own protocol and use it, and self-healing, i.e. the ability to recompute a new protocol if something goes wrong, so that you can continue interaction.

Currently if I’m using a particular protocol like SSL, it will start off by negotiating the particular ciphers that we want to use, the version of the protocol, and so on. In some cases we may be willing to negotiate a particular protocol, but it’s always going to come from a fixed menu. So the participants, Harry and Sally, have a fixed menu of protocols that they’re willing to use, and Harry says to Sally, “which protocol do you want to use when you interact?” And the problem is that if they if can’t agree on one of the fixed pre-programmed protocols then they get no service. Also fixed protocols are based on a fixed set of requirements and assumptions, and so on, so we have very limited reconfiguration in response to any environment changes. If the key gets compromised, then we can probably
react to that by re-keying, but there are a number of other things that can go
wrong which we can’t react to, because we’re just using a fixed set of protocols
with these fixed assumptions.

Matt Blaze: Quick clarification, are you using the word protocol in the usual
meaning or is there some subtle distinction between a protocol and its parame-
ters?

Reply: I’m using it in the usual sense: the description of the flows as opposed
to just the configuring part. Most existing protocols won’t even allow you to
change the configuration, except in some limited cases, although sometimes you
can use different versions of a particular protocol, in effect using sub-protocols
of some larger protocol.

One of the few architectures that might provide some kind of self-configu-
ration is protocol-agnostic Jini. Here Sally first of all registers the protocol with
the Jini server, and in effect she’s saying, if anybody wants to talk to me, then
the protocol they need to use is going to be available from the server. Jini has
been used in systems which people would claim to be self-configuring, self-healing
systems. However, the difficulty is that Sally is just registering a single fixed pro-
tocol. But perhaps Harry has his own particular collection of requirements, and
he doesn’t want to follow Sally’s protocol because it’s not got certain properties
that he’s interested in.

So we want to provide some kind of framework so that complete strangers
can negotiate and generate suitable protocols on the fly. These protocols are
going to have to be self-configuring, they’re going to have to figure out what
message exchanges are going to be necessary to keep the parties involved happy.
And we want them to be self-healing, so that if something happens at one of
the principals that they decide they don’t like, or if they don’t trust certain
aspects of particular principals anymore, then perhaps they want to go away
and renegotiate a new protocol.

So obviously what we’re going to be doing is synthesising a protocol from
the collection of assumptions about the principals, and a collection of goals. The
goals are what it is we want this protocol to do before we’re willing to interact
with somebody. And so we get a collection of assumptions about keys, princip-
als, and so forth, we have our goals which are the requirements for our final
interaction, and what we’re going to do is automatically synthesise a protocol
on the fly. The architecture works something like this: Harry and Sally both
have their own assumptions and their own goals, Sally has her goals which are,
‘for me to provide service to somebody this is what I expect before we start
interacting.’ When Harry contacts Sally and says, “I’m Harry,” Sally then pro-
vides her assumptions and goals to Harry. Harry has then got to go away and
synthesise the protocol using his own assumptions and goals plus those of Sally.
And having synthesised the protocol he would send his protocol plus his own
assumptions to Sally so that she can then go ahead and verify the protocol given
the assumptions and her goals, and if everybody’s happy then they will begin
to install and then execute the protocol. So in very, simple terms that’s how the
synthesis should work.