Int J Inf Secur (2005) 4: 17–28 / Digital Object Identifier (DOI) 10.1007/s10207-004-0047-7

# An overview of the verification of SET

**Giampaolo Bella**[1], **Fabio Massacci**[2], **Lawrence C. Paulson**[3]

[1] Dipartimento di Matematica e Informatica, Università di Catania, Viale A. Doria 6, 95125 Catania, Italy
e-mail: giamp@dmi.unict.it
[2] Dipartimento di Informatica e Telecomunicazioni, Università di Trento, Via Sommarive 14, 38050 Povo (Trento), Italy
e-mail: massacci@ing.unitn.it
[3] Computer Laboratory, University of Cambridge, J J Thomson Avenue, Cambridge CB3 0FD, UK
e-mail: lcp@cl.cam.ac.uk

Published online: 25 January 2005 – 

**Abstract.** This paper describes the verification of Secure Electronic Transaction (SET), an e-commerce protocol by VISA and MasterCard. The main tasks are to comprehend the written documentation, to produce an accurate formal model, to identify specific protocol goals, and, finally, to prove them. The main obstacles are the protocol's complexity (due in part to its use of digital envelopes) and its unusual goals involving partial information sharing. Our verification efforts show that the protocol does not completely satisfy its goals, although the flaws are minor. The primary outcome of the project is experience with verification of enormous and complicated protocols. This paper summarizes the project – the details appear elsewhere [11–13] – focusing on the issues and the conclusions.

**Keywords:** Automated reasoning – Network protocols – Computer security – Formal verification – Secure Electronic Transactions (SET)

## 1 Introduction

Recent years have seen substantial progress in the formal verification of security protocols. Detailed analysis of cryptographic primitives, verification of Internet standards, and substantial progress in the automation of model-checking and theorem-proving procedures for security verification have boosted a field that outsiders believe is populated only by "Yet-Another-Look-at-Needham–Schroeder" papers.

Protocol verification techniques fall into several categories. A general-purpose model checker can verify protocols, as pioneered by Lowe and colleagues at Oxford [23, 40]. A general-purpose proof tool can also be effective, as in Paulson and colleagues' work based on higher-order logic and the tool Isabelle [9, 33], and in Schneider's work based on process algebras and the tool PVS [41].

Additionally, there exist several specialized protocol analysis tools. Most perform an exhaustive search in the spirit of model checking. For instance, we can use Athena by Song [43] which is based on the strand space model, the process algebra model-checking tool developed by Focardi and Gorrieri [38], tools based on rewriting proposed by Rusinowitch and his group [19, 21], systems that use reduction to planning via satisfiability such as the AVISS tool [3, 4] or via logic programming as proposed by Carlucci and Massacci [18]. Among the model-checking tools, the most interesting uses on-the-fly model checking with infinite-state automata [3, 7, 8]. Among the specialized tools having deductive capabilities, the best is Meadows' NRL [29]. Also, Brackin has used a higher-order implementation of BAN logics for the specification of protocols and can verify authentication properties [16]. Cohen's TAPS processes the protocol specification and verifies the desired properties using a resolution theorem prover [20]. Computer-assisted methods based on static analysis have only recently been developed as in the work by Blanchet et al. [1, 15]. For additional sources the reader may consult Meadows's recent and fairly exhaustive survey [30].

Formal proof is preferable for establishing properties, while model checking is best for finding attacks. Exhaustive search is only feasible if the model is kept as small as possible, for example by minimizing the number of permitted executions. If the assumptions are too strong, the absence of an attack does not guarantee correctness. Interactive proof tools are not automatic, but they offer flexibility in expressing specifications and proofs. Models need not be finite and can therefore be more realistic.

Many practical industrial protocols have been formally verified using interactive or semi-interactive proof tools. Protocols like Kerberos IV [14], the Internet Key Exchange protocol [29], the Cybercash protocol [24] and the TLS/SSL protocol [34] have all yielded to automatic

or semi-automatic tools. One particular protocol has proved to be particularly resistant to verification: the SET (Secure Electronic Transaction) protocol by VISA and Mastercard.

SET [26–28] has been proposed by a consortium of credit card companies and software corporations to secure e-commerce transactions. When a customer makes a purchase, the SET protocol guarantees authenticity of the transaction while keeping the customer's account details secret from the merchant and his choice of goods secret from the bank. Its appeal to researchers working in verification is the possibility of demonstrating that one's own verification technology is mature enough to cope with the demands of a huge, complex, industrial protocol.

Indeed, many researchers have worked on the problem: for instance, Meadows and Syverson [31] have proposed a language for describing SET specifications but have not actually verified the protocol. Kessler and Neumann [22] have extended an existing belief logic with predicates and rules to reason about *accountability*. Although accountability is not a stated goal of SET, it is clearly desirable. Kessler and Neumann concentrate upon the merchant's ability to prove to a third party that the order information originated with the cardholder. Using the calculus of the logic, they conclude by pen and paper that the goal is met, so the cardholder cannot repudiate the transaction. Stoller [44] has proposed a theoretical framework for the bounded analysis of e-commerce protocols but has only considered an overly simplified description of the payment protocol of SET. Hui and Lowe [24] have proposed a general theory to transform a complex protocol into a simpler protocol while preserving any faults. However, they have limited their actual analysis to the Cybercash protocol. The claim "we plan to apply our verification technology to SET" was a frequent conclusion to talks and papers at the end of the millennium. Yet, the protocol has resisted most verification attempts.

Why is SET such a challenge for formal verification? The first obstacle is its documentation [25–28], which fills over 1000 pages. However, the main obstacle is the protocol itself. Protocols proposed in scientific journals are typically short, straight-line programs: they seldom go beyond two levels of encryption and generate few secrets. Even more sophisticated protocols such as Optimistic Fair Exchange [5] or Group Protocols [6] can be described in a few pages and have been successfully verified [36, 42]. Internet protocols such as IKE and TLS use cryptography rather sparingly compared to SET. SET has many features that make its verification unusual and hard.

The complex structure of SET makes it a benchmark for security protocol design and verification. Whether SET is a success or not, it is undoubtedly a real-world protocol, arising from industry. Such a gigantic protocol cannot be convincingly verified without tool support. However, tools require formal models. Even the task of designing an adequate formal model may be too much for human intuition.

We succeeded in analysing an abstract, but still highly complex, version of SET: the registration phases [12] and the purchase phase [11]. The difficulty consisted in digesting the specification and scaling up. This is a major result: our method scales to a level of complexity where intuition falters. Unfortunately, we discovered that the method, based on human interaction with a semi-automatic but powerful prover, has reached a point where the complexity of the proofs and the sheer size of the intermediate properties will require further advances to scale further.

The paper begins by outlining the SET protocol (Sect. 2). It briefly introduces the inductive approach and Isabelle (Sect. 3). It outlines our proofs of the registration phase (Sect. 5) and the purchase phase (Sect. 6) of SET. Finally, there are some general conclusions (Sect. 7).

## 2 The SET protocol

People today pay for online purchases by sending their credit card details to the merchant. A protocol such as SSL or TLS keeps the card details safe from eavesdroppers but does nothing to protect merchants from dishonest customers, or vice versa. SET addresses this situation by requiring cardholders and merchants to register before they may engage in transactions. A cardholder registers by contacting a certificate authority, supplying personal account details and his proposed signature verification key (the public half). Registration allows the authorities to vet an applicant, who if approved receives a certificate confirming that his public key has been registered. All orders and confirmations bear digital signatures, which provide authentication and could potentially help to resolve disputes.

A SET purchase involves three parties: the cardholder $C$, the merchant $N$, and the payment gateway $PG$ (loosely speaking a bank). The cardholder shares the order information $OI$ with the merchant but not with the payment gateway. He shares the payment information $PI$ with the bank but not with the merchant. A SET *dual signature* accomplishes this partial sharing of information: the cardholder makes separate hashes of the order information and the payment information and signs the pair of hashes. Every other party receives the hash of the withheld information and the signature of the pair. Each party can confirm that the hashes in their possession agree with the hash signed by the cardholder. In addition, cardholder and merchant compute equivalent hashes for the payment gateway to compare. He confirms their agreement on the details withheld from him. Figure 1 shows an abstract dual signature.

All parties are protected. Merchants do not normally have access to credit card numbers. Moreover, the mere possession of credit card details does not enable a criminal to make a SET purchase: he needs the cardholder's signature key and a secret number that the cardholder receives upon registration. (Some optional features of

$$\begin{aligned} C \longrightarrow M &: OI, \mathsf{Hash}(PI), \{PI\}_{pubK_{PG}}, \\ &\quad \{\mathsf{Hash}(OI), \mathsf{Hash}(PI)\}_{signK_C} \\ M \longrightarrow PG &: \mathsf{Hash}(OI), \{PI\}_{pubK_{PG}}, \\ &\quad \{\mathsf{Hash}(OI), \mathsf{Hash}(PI)\}_{signK_C} \end{aligned}$$

**Fig. 1.** A dual signature

SET – presumably demanded by commercial interests – weaken these properties. A merchant can be authorized to receive credit card numbers and has the option of accepting payments without digital signatures.)

SET is a family of protocols. The five main ones are cardholder registration, merchant registration, purchase request, payment authorization, and payment capture. There are many additional minor protocols, for example to handle errors. SET is enormously more complicated than SSL, which merely negotiates session keys between the cardholder's and merchant's Internet service providers.

Let us briefly review SET's interesting features:

- Security bootstrapping is unusual: the initiator possesses no digital proof of identity and authenticates himself by filling in a registration form whose format is not specified. Authentication takes place outside the protocol, when the cardholder's bank examines the completed form.
- The protocol uses multiple nested encryptions and several message fields. These require abbreviations, make the manual unwinding of the specifications impossible and restrict analysis to tools supporting equational reasoning.
- SET uses *digital envelopes*. A digital envelope consists of two parts: one, encrypted using a public key, contains a fresh symmetric key $K$ and identifying information; the other, encrypted using $K$, conveys the full message text. Digital envelopes keep public-key encryption to a minimum, but the symmetric keys complicate the reasoning. It hampers the usual model-checking technique to limit the state space (limiting different keys and nonces to a handful) as it would not even allow a single execution to complete, let alone two or more parallel ones.
- The goal of the protocol is to protect the information about merchandise from the bank and the information about credit from the merchant while authenticating the entire transaction. The partial sharing of information among the three peers leads to unusual protocol goals.
- SET has many alternative protocol paths that make it impossible to single out the few key roles used either by manual analysis (as in the strand space model) or by model checkers to restrict the search space.

Are these features or bugs? Though some security experts may claim that SET is a monster designed by a committee, others will note that many of these features are valuable. Alternative protocol paths are necessary in practice in order to allow diversity. Security-aware customers may have pre-registered with a financial institution and thus secured their credit cards from the merchant's eyes. Other customers may decide to trust the merchant and thus be content with a transaction secured against the outside world. From a merchant's perspective, all customers should be able to conclude a purchase, whether they have bothered to pre-register or not.

This paper is intended to summarize our work on the SET protocol: the issues, the results and the lessons learned. Detailed descriptions of the verification are published elsewhere [11–13].

## 3 Isabelle and inductive protocol verification

We used the Isabelle theorem prover with the inductive approach to protocol verification, building on previous experience on a wide range of protocols, including industrial ones such as Kerberos [14] and TLS [34].

Isabelle/HOL [32] is an interactive proof tool for higher-order logic. Isabelle provides a simplifier, a predicate calculus theorem prover, a choice of proof languages, and automatic generation of LaTeX documents. Isabelle's support for inductive definitions is particularly strong, both in its specification language and in its prover automation. However, other tools for higher-order logic could be suitable, provided they fully support conditional equational reasoning.

The inductive approach [33] verifies protocols using the standard techniques of operational semantics. An inductive definition defines the possible executions of a system consisting of the honest protocol participants and an active attacker, the Spy. An execution comprises any number of attempted protocol runs and is a trace of message transmissions and other events.

Authentication and agreement are expressed using safety properties over traces and proved by induction over traces. We typically prove that any trace containing a particular event $x$ must also contain some other event $y$.

Secrecy properties are hardest to prove. If we are concerned with the secrecy of a certain key $K$, then we must prove $K \neq K'$ for each key $K'$ that might be compromised. Every encrypted message produces a case split, since we must prove that $K$ is secure whether or not the encrypting key is secure. Protocols with many steps or many options can generate a huge number of cases. Despite the difficulties, we can use established techniques and tools to prove secrecy.

Most protocols, even esoteric ones like non-repudiation and fair exchange protocols, involve the standard cast of characters: Alice, Bob, possibly Charlie, and a trusted third party. SET is different: it has cardholders, merchants, payment gateways and a hierarchy of certificate authorities. Still, changing Isabelle's theory of protocols to use SET's cast of characters was easy.

The model includes a set of honest agents, whose long-term keys can never become compromised. (Arguably,

our model is too optimistic.) For typical protocols, where long-term keys are never transmitted, proving that they remain secure is trivial. The Spy controls another set of agents, with full access to their internal states. The Spy also controls the network and retains every transmitted message. Session keys may become compromised, for example if they are sent to compromised agents.

A standard theory of messages and their constructors underlies these inductive models. Messages in our model form a recursive datatype (equivalently, a free algebra). A nonce can never equal an agent name or a session key, for instance. Such assumptions are more realistic than one might expect: different kinds of items are likely to have different lengths and even a different bit-wise encoding. Concatenation of messages may seem to be inherently associative. However, the ISO-DER encoding of a sequence of six random numbers has a bit-wise encoding different from the concatenation of a pair of sequences of three numbers.

Encryption is injective in our theory. Only one key can decrypt a ciphertext, which can yield only one plaintext. This assumption is plainly false for low-level applications of encryption, where using the wrong key yields a plaintext of random bits. However, it is correct provided "each encrypted message contains sufficient redundancy to allow a principal who decrypts it to verify that he has used the right key", to quote Burrows et al. [17, p. 237]. Most research on protocol verification relies on this assumption.

Our model does not allow reasoning about exclusive-OR (XOR for short). XOR breaks down our representation of messages as a free algebra, since it satisfies several equations such as associativity, commutativity and self-cancellation. Intuitively, the problem is that the XOR of two messages can potentially yield a message of any form, though such a possibility is highly unlikely in practice. One needs substantial logical complications to make the highly unlikely formally impossible while keeping the rather unlikely still possible.

- XORing two compromised 1024-bit private keys should not yield an uncompromised key.
- In contrast, XORing together a long chain of pairwise XORed nonces (each of the form $N_i \oplus N_{i+1}$), should reveal an initially uncompromised nonce $N_0$ because of the cancellation law.

An appealing theory of XOR has not been proposed so far in the literature, and its invention would not be justified in this case. Fortunately, SET uses XOR only in one place: at the end of cardholder registration, to compute the so-called PANSecret. Rather than proving the secrecy of the PANSecret, we prove the secrecy of the two random numbers used in its calculation, substantially yielding the same desired effect. Proving the secrecy of the PANSecret would require additional assumptions in order to exclude the possibility that the XOR could yield a compromised secret. Still, as discussed in Sect. 5.4 below, SET's use of XOR introduces a vulnerability.

## 4 Modelling issues

Researchers compete to produce the fastest automatic tools. However, the main obstacle to protocol verification lies in digesting the documentation and producing a formal model. Understanding hundreds of pages of text is a massive undertaking. Meticulous care is essential to avoid defining an incorrect model.

The main SET documents are the *Business Description* [26], the *Programmer's Guide* [28], and the *Formal Protocol Definition* [27]. SET is defined using Abstract Syntax Notation One (ASN.1).[1] The *Programmer's Guide* presents each message format as a figure based on the underlying ASN.1 definition, augmented with a detailed English description of how to process each message. The *Formal Protocol Definition* consists of the *Programmer's Guide* with the ASN.1 notation inserted and the English text removed. Since the ASN.1 adds little to the figures, the formal protocol definition essentially consists of syntax without semantics. It describes the message formats but says nothing about how messages are processed. For that information, we had to rely on the *Programmer's Guide*.

The enormous size and complexity of the SET message formats demanded simplification. As we have discussed elsewhere [10, 13], this was not always straightforward, forcing us to decide what constituted SET's core feature set. We eliminated payment by installments (since it can be modelled by repeated transactions) and modelled only authorized transactions (so unauthorized transactions were modelled by silent denial). Other researchers can make other choices.

Attacks against protocols often arise from unclear assumptions about the operating environment rather than from actual protocol flaws. Experts can dispute whether the formal model accurately reflects the real world and thus whether the attack is realistic. Consider Lowe's famous attack [23] against the Needham–Schroeder public-key protocol: Alice talks to Charlie, who happens to be dishonest and proceeds to fool Bob. In this scenario, Charlie is a dishonest insider. However, Needham and Schroeder designed the protocol to protect insiders from outsiders with the express assumption that honest insiders will only talk to other equally honest insiders.

SET has a much more complex environment, and parts of its operation are specifically left "out of band". Our formal model has to make reasonable assumptions about these parts, which are sketched in the *SET External Interface Guide* [25]. It also must specify which insiders can be compromised and innumerable other details. It also has to define the protocol goals since the documentation outlines them only in general management terms.

[1] `http://www.asn1.org`

## 5 Verifying the registration protocols

The cardholder registration protocol (Fig. 2) comprises three message exchanges between the cardholder and a certificate authority. In the first exchange, the cardholder requests registration and is given the certificate authority's public keys. In the second exchange, the cardholder supplies his credit card number, called the PAN, or Primary Account Number; he receives an application form suitable for the bank that issued his credit card. In the third exchange, the cardholder returns the completed application form; in addition, he delivers his public signature key and supplies a 20-byte secret number (the CardSecret). Finally, the cardholder receives a certificate that contains his public signature key and another 20-byte secret number, the PANSecret. The registration protocol for merchants is simpler: it has only two message exchanges and involves no credit card number.

Conceptually, cardholder registration is straightforward. Its chief peculiarities are that the cardholder is authenticated by the registration form containing the PAN, not by the knowledge of a secret key, and that long-term keys can be created on the fly. The first point is critical for modelling and is discussed elsewhere [13]. The second point makes verification difficult.

### *5.1 Dynamic creation of long-term keys*

Typical modelling of a public/private key pair associates each half to the agent holding it: there is a function mapping each agent's name to his public key. Thus each agent has precisely one public key, and therefore one private key. This greatly simplifies all secrecy proofs.

– If the agent has only one key and he is not compromised, his private key is by definition not compromised at the beginning of the protocol. If the private key is never sent in a message – and this is true of most protocols – it will never be compromised. If the agent is compromised, his keys are lost from the start. This modelling of key compromise avoids certain case splits in secrecy proofs.
– If the agent is not compromised, then the Spy cannot read messages encrypted using his public key. We can focus on the remaining trace and apply the inductive hypothesis.

A similar phenomenon takes place in model-checking approaches. There is no search involved in asymmetric cryptography when key pairs are assigned to agents from the outset.

If agents have more than one private key and can generate public/private key pairs on the fly, it is necessary to have a complex background theory on keys which formally states all obvious properties.

– If agents can generate keys on the fly, one should not be able to generate existing keys by chance, a public half of a key should not be be equal to a secret half of another asymmetric key or to a symmetric key, there should be an infinite supply of keys, etc.
– In proofs, each time a public key is used we can no longer look at the agent and conclude whether or not the encrypted message is compromised. Case splits would arise: one particular key could be compromised, or a cardholder could use a key that is not compromised before step $i$ but becomes compromised soon afterwards. Also, a cardholder could, but is not obliged to, register a fresh key each time, and a cardholder using stale keys would generate additional case splits.

This uncertainty makes proofs hard and model checking impossible.

Indeed, in our first model of the cardholder registration protocol presented at ESORICS [13], we modelled these possibilities successfully. In the version described in the JSAC paper [12], we have reverted to the standard modelling approach, where one key pair for signature and one for encryption are syntactically associated to each agent, because it is more readable.

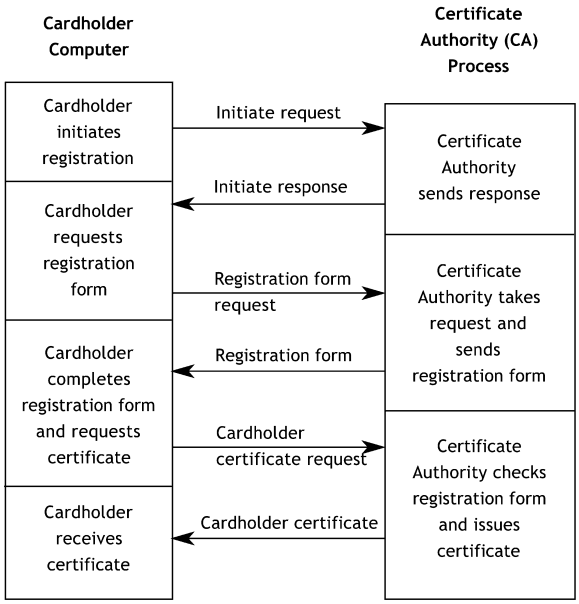



**Fig. 2.** Cardholder registration

### *5.2 Key dependency chains*

Another obstacle to verification – especially proving secrecy – is SET's heavy use of digital envelopes. Digital envelopes can generate a trace where in message 1 there is a key encrypting a key for message 2, and so forth. To prove the secrecy of the last key, one must prove the secrecy of all dynamically created keys in the chain.

In most formal analysis of key distribution protocols, the initial public keys are assigned to agents from the outset. The final step of actually encoding data with the distributed session key is never modelled, as one just proves that the session key remains secret. So a dependency chain of length two in the real protocol is actually cut

down to no chain in the modelled protocol. Yahalom [35] and Kerberos [14] have a (modelled) dependency chain of length one: one session key encrypts just one secret. Also, protocols such as IKE have a short dependency chain: the public and private keys used during phase 1 are just used to encrypt the session key which will be used during phase 2 of the protocol.

The explicit modelling of digital envelopes and the usage of digital envelopes to transmit dynamically created secrets produce long formal dependency chains. With SET, the dependency chain has length two, or three if signing keys are generated dynamically. It may not sound like much of an increase, but the jump from zero or at most one to two or three requires new proof techniques. This is a major problem because digital envelopes are ubiquitous in any practical protocol that uses public key cryptography to ensure secrecy. Indeed, if one sees a specification of a protocol that uses public-key cryptography for encryption but does not use digital envelopes, it is probably because it abstracts them away, not because they are not there.

To cope with arbitrary dependency chains, we generalized the technique used for Yahalom [35] and Kerberos [14]. We defined a transitive relation specifying where in a given trace the loss of one key leads to the loss of another: when the first key was used to encrypt the second key in some messages sent during that trace of events. This creates a dependency relation between the second key and the first key. Then, we proved some lemmas that rule out dependencies or bound what can be lost.

- The simplest dependencies that can be ruled out are those depending on the cardholder's public key. No key depends on the cardholder's secret key because no key is ever encrypted with the cardholder's public key.
- The secrecy of a key never used in a trace cannot depend on the secrecy of another key previously used in the trace. Also, if unused keys are lost to the Spy, then they must be held by compromised agents.

It is also possible to give a protocol-independent treatment of dependency chains. The generic Isabelle theory of protocol messages defines a relation yielding the keys necessary to decrypt some message belonging to a given set of messages. The definition is independent of any particular protocol. After extending the message theory with SET constructs, we have used the notion extensively in the proofs about SET.

However, the proofs for protocols as complex as SET often have many intermediate subgoals that span many pages. If we define the key dependency chain specifically for the protocol under verification, then we can avoid some case splits by construction. We define the relation to refer to the specific protocol steps that produce the dependency chain. The other protocol steps, no matter how complicated, are ruled out by construction.

This treatment of the relation is practical and still safe, since the proofs will reveal any errors. If our relation omits some dependency, then our lemma bounding the possible losses will be useless for proving other secrecy theorems. Moreover, the case of the secrecy theorem that we are unable to prove will indicate which dependency was missed.

In the current model of cardholder registration, the chain links only three items: two symmetric keys and one nonce. If asymmetric keys can be generated on the fly, the chain can become longer and the bounding lemmas more complex. Having both on-the-fly generation of asymmetric keys and digital envelopes adds more than the sum of their complexities.

### *5.3 Modelling the fifth message of registration*

Let us consider these points more precisely. Here is the fifth message, *Cardholder Certificate Request*:

$$\begin{aligned}&5.\ C \to CA : \mathsf{Crypt}_{\mathrm{KC3}}(m,\ S),\\ &\qquad\qquad \mathsf{Crypt}_{\mathsf{pubEK}\ CA}(\mathrm{KC3}, \mathrm{PAN}, \mathrm{CardSecret})\\ &\text{where } m = C,\ \mathrm{NC3},\ \mathrm{KC2},\ \mathsf{pubSK}\,C\\ &\text{and } S = \mathsf{Crypt}_{\mathsf{priSK}\,C}(\mathsf{Hash}(m, \mathrm{PAN}, \mathrm{CardSecret}))\end{aligned}$$

The cardholder chooses an asymmetric signature key pair. He gives the public key, $\mathsf{pubSK}\,C$, and the number CardSecret to the certificate authority. This message is a digital envelope, sealed using the key KC3; it contains another key, KC2, which the certificate authority uses for encrypting the *Cardholder Certificate*:

$$\begin{aligned}&6.\ CA \to C : \mathsf{Crypt}_{\mathrm{KC2}}\\ &\qquad\qquad (\mathrm{Sign}\ CA\ (C, \mathrm{NC3},\ CA, \mathrm{NonceCCA}),\\ &\qquad\qquad\ \mathsf{Cert}_{CA}(\mathsf{pubSK}\,C, \mathrm{PANSecret}),\\ &\qquad\qquad\ \mathsf{Cert}_{\mathsf{RCA}}(\mathsf{pubSK}\ CA)))\\ &\text{where PANSecret} = \mathrm{CardSecret} \oplus \mathrm{NonceCCA}\end{aligned}$$

The certificate authority returns a certificate for the cardholder's public signature key. The certificate also includes the cryptographic hash of PANSecret. This 20-byte number is the XOR of the CardSecret and NonceCCA: a nonce chosen by the certificate authority. The purpose of these nonces is twofold: CardSecret will be used by the cardholder to confirm purchases on top of the digital signature (the hash of CardSecret must be added for each payment instruction), and PANSecret as a whole will be used to generate the cardholder's name in the X.509 certificate format. The name of the public-key holder in the certificate will not be $C$ but $\mathsf{Hash}$(PANSecret, PAN).

*Remark.* The secrets KC3, KC2, NonceCCA form a dependency chain, requiring the new proof technique mentioned above. Removing the digital envelopes here would shorten the dependency chain – as would disposing with NonceCCA, as we recommend below.

Figure 3 presents the Isabelle specification of message 5. It is hard to read, but comparing it with the informal notation above conveys an idea of the syntax. The inductive definition consists of one rule for each protocol message, which extends a given trace. (Note that `#` is Isabelle syntax for the list "cons" operator. In the rule for message 5, the current trace is called *evs5*.) One of the rule's preconditions is that *CardSecret* must be fresh:

*Nonce CardSecret* $\notin$ *used evs5*

The nonce *NC3* and the two symmetric keys (*KC2* and *KC3*) must also be fresh. Other preconditions check that the cardholder has sent an appropriate instance of message 3 to the certificate authority and has received a well-formed reply. If the preconditions are satisfied, then *C* can generate the corresponding instance of message 5.

### *5.4 Security of the PANSecret*

We did not discover any attacks against cardholder registration. However, we did discover a modification that would improve the protocol. Under reasonable assumptions, the PAN, PANSecret and other sensitive information remain secure. Among the reasonable assumptions is that certificate authorities are not compromised. (Though this might be argued about a financial institution as such, it might be false about the institution's outsourced software!) Here is a flaw: the PANSecret is computed as the XOR of CardSecret and NonceCCA, which gives the certificate authority full control over its value. One would like to be able to trust the certificate authorities, but banks have issued insecure personal information numbers.

> "One small upper-crust private bank belied its exclusive image by giving all its customers the same PIN. This was a simple programming error; but in another, more down-market institution, a programmer deliberately arranged things so that only three different PINs were issued, with the idea that this would provide his personal pension fund" [2, p. 35].

The remedy is trivial: compute the PANSecret by hashing instead of XOR. Another remedy is to leave its choice entirely to the cardholder's computer – after all, it exists for the cardholder's protection. If two nonces are needed, one (PANSecret) disclosed to the payment gateway and another (CardSecret) disclosed only to a certificate authority, then let the cardholder generate both of them.

This modification would eliminate NonceCCA, and with it the need to encrypt message 6, which would contain only public-key certificates. We could dispense with the key KC2 and eliminate the dependency chain KC3, KC2, NonceCCA. These changes would make the protocol simpler and more secure against a compromised certificate authority.

## 6 Verifying the purchase protocols

A SET purchase can involve three protocols: purchase request, payment authorization, and payment capture.

```
⟦evs5 ∈ set_cr;  C = Cardholder k;
 Nonce NC3 ∉ used evs5; Nonce CardSecret ∉ used evs5;
 NC3 ≠ CardSecret;
 Key KC2 ∉ used evs5; KC2 ∈ symKeys;
 Key KC3 ∉ used evs5; KC3 ∈ symKeys; KC2≠KC3;
 Gets C {|sign (invKey SKi) {|Agent C, Nonce NC2, Nonce NCA|},
          cert (CA i) EKi onlyEnc (priSK RCA),
          cert (CA i) SKi onlySig (priSK RCA)|}
      ∈ set evs5;
 Says C (CA i)
      {|Crypt KC1 {|Agent C, Nonce NC2, Hash (Pan (pan C))|},
        Crypt EKi {|Key KC1, Pan (pan C),
                             Hash {|Agent C, Nonce NC2|}|}|}
      ∈ set evs5⟧
⟹ Says C (CA i)
         {|Crypt KC3
           {|Agent C, Nonce NC3, Key KC2, Key (pubSK C),
             Crypt (priSK C)
               (Hash {|Agent C, Nonce NC3, Key KC2,
                       Key(pubSK C), Pan(pan C), Nonce CardSecret|})|},
           Crypt EKi {|Key KC3, Pan (pan C), Nonce CardSecret|}|}
    # evs5 ∈ set_cr
```

**Fig. 3.** Cardholder registration in Isabelle (message 5)

The first two of these often behave as a single protocol, which is how we model them. (We have not investigated payment capture.) The protocol is too complex to present here in full. Even the means of identifying the transaction is complicated. The cardholder and merchant may each have an identifying number; sometimes a third number is chosen. The choice of method is actually left open by SET designers. These options are clearly derived from the world of paper transactions ("Our ref: ... ", "Your ref: ... ", "Invoice num: ... ", etc.) and make business sense in a commercial context.

From the point of view of verification, this lack of clarity is not acceptable. To prove any result about the authenticity, integrity or secrecy of the transaction details at the end of the protocol, we must clearly identify the transaction at the beginning of the protocol. For sake of simplicity, we discard all but one of the identification options: we use the merchant's transaction identifier. This option – sanctioned in the *SET External Interface Guide* [25] – corresponds to a widespread practice for tracking orders: browser cookies.

The essential parameters of any transaction are the *order description* (presumably a text string) and the *purchase amount*, which is part of the overall *payment instructions*. The cardholder forms a dual signature on the order information and payment information, as outlined in Sect. 2, and sends it to the merchant. The merchant forwards the payment information, under his signature, to the payment gateway. Only the payment gateway can read the account details, which include the PAN and the PANSecret. If they are acceptable, he replies to the merchant, who confirms the transaction with the cardholder.

Other details of our model include an event to model the initial shopping agreement, which lies outside SET. Our model also includes the possibility of unsigned purchases. These allow unregistered cardholders to use SET using a credit card number alone and offer little protection to merchants. SET perhaps offers this option in order to provide an upgrade path from SSL.

An example illustrates the complexity of the dual signature. Message 3 is the actual purchase request from the cardholder to the merchant:

3. $C \to M$ : PIDualSign, OIDualSign

Here, the cardholder $C$ has computed

$$
\begin{aligned}
\mathrm{HOD} &= \mathsf{Hash}(\mathrm{OrderDesc}, \mathrm{PurchAmt}) \\
\mathrm{PIHead} &= \mathrm{LID\_M}, \mathrm{XID}, \mathrm{HOD}, \mathrm{PurchAmt}, \mathrm{M}, \\
&\quad \mathsf{Hash}(\mathrm{XID}, \mathrm{CardSecret}) \\
\mathrm{OIData} &= \mathrm{XID}, \mathrm{Chall\_C}, \mathrm{HOD}, \mathrm{Chall\_M} \\
\mathrm{PANData} &= \mathrm{PAN}, \mathrm{PANSecret} \\
\mathrm{PIData} &= \mathrm{PIHead}, \mathrm{PANData} \\
\mathrm{PIDualSign} &= \mathsf{Sign}_{\mathsf{priSK}\,C}(\mathsf{Hash}(\mathrm{PIData}), \mathsf{Hash}(\mathrm{OIData})), \\
&\quad \mathsf{Crypt}_{\mathsf{pubEK}\,P}(\mathrm{PIHead}, \mathsf{Hash}(\mathrm{OIData}), \mathrm{PANData}) \\
\mathrm{OIDualSign} &= \mathrm{OIData}, \mathsf{Hash}(\mathrm{PIData})
\end{aligned}
$$

LID_M and XID are unique (but guessable) transaction identifiers generated by the merchant's software; Chall_C and Chall_M are nonces; the remaining fields are all derived from PAN, PANSecret and CardSecret.

Because of the hashing, all the information appears repeatedly. Although in the real world the hash of any message is a short string of bytes, in the formal model the hash of message $X$ is literally $\mathsf{Hash}\,X$: a construction involving $X$. The formal model of message 3 involves massive repetition. Most digital envelopes involve hashing, causing further repetition. Figure 4 presents this message using Isabelle syntax.

The SET documentation does not specify what operational properties to prove. The Business Description only offers the following requirements [26, p. 6]:

> SET addresses seven major business requirements:
>
> 1. Provide confidentiality of payment information and enable confidentiality of order information that is transmitted along with the payment information.
> 2. Ensure the integrity of all transmitted data.
> 3. Provide authentication that a cardholder is a legitimate user of a branded payment card account.
> 4. Provide authentication that a merchant can accept branded payment card transactions through its relationship with an acquiring financial institution.

We dropped requirements such as using best practice or providing interoperability.

These high-level goals can be mapped in a variety of ways into operation goals, and we used our reading of the entire Business Description and the Programmer's Guide to define them. Here are two examples:

- Obviously, the PAN and PANSecret must remain secure. Each party to a purchase must be assured that the other parties agree on all the essential details: the purchase amount, the transaction identifier, the order description and the names of the other agents.
- The payment gateway must be assured that he has received the right payment information and the order description is correct, even though the order information is withheld from him.

We proved most of these properties. Some proofs were easy; few theorems had intrinsically difficult proofs. The sheer number of theorems and supporting lemmas was an obstacle – many results had separate versions for signed and unsigned purchases. The complexity of theorem statements, caused by the complicated SET message formats, was an obstacle. Not knowing precisely what to prove was a major obstacle: if we had problems proving an assertion, we had to decide whether to weaken it somehow, to look harder in the SET documentation for some omitted field, or try harder with the proof itself.

When the merchant sees a dual signature in a purchase request, he is assured that it originated with the

```
⟦evsPReqS ∈ set_pur;  C = Cardholder k; CardSecret k ≠ 0;
 Key KC2 ∉ used evsPReqS;  KC2 ∈ symKeys;
 Transaction = {|Agent M, Agent C, Number OrderDesc, Number PurchAmt|};
 HOD = Hash{|Number OrderDesc, Number PurchAmt|};
 OIData = {|Number LID_M, Number XID, Nonce Chall_C, HOD, Nonce Chall_M|};
 PIHead = {|Number LID_M, Number XID, HOD, Number PurchAmt, Agent M,
           Hash{|Number XID, Nonce (CardSecret k)|}|};
 PANData = {|Pan (pan C), Nonce (PANSecret k)|};
 PIData = {|PIHead, PANData|};
 PIDualSign = {|sign (priSK C) {|Hash PIData, Hash OIData|},
                EXcrypt KC2 EKj {|PIHead, Hash OIData|} PANData|};
 OIDualSign = {|OIData, Hash PIData|};
 Gets C (sign (priSK M){|Number LID_M, Number XID,
                         Nonce Chall_C, Nonce Chall_M,
                         cert P EKj onlyEnc (priSK RCA)|})
   ∈ set evsPReqS;
 Says C M {|Number LID_M, Nonce Chall_C|} ∈ set evsPReqS;
 Notes C {|Number LID_M, Transaction|} ∈ set evsPReqS⟧
⟹ Says C M {|PIDualSign, OIDualSign|} # evsPReqS ∈ set_pur
```

**Fig. 4.** The signed purchase request message

cardholder. The formal proof, like the intuitive one, argues that only the cardholder knows his private signature key. The proof uses induction, as usual, and applies three easily proved technical lemmas. This theorem is important: by verifying the dual signature, specifically the transaction identifier XID, the merchant can be assured that he and the cardholder agree on the details of the purchase.

The theorem described above is a typical agreement guarantee. The agreement guarantees between other pairs of agents are also easy to prove. The total effort, due to the obstacles mentioned above, is considerable.

Secrecy properties are typically harder to prove than other properties, and this is also true for SET. Unless equational rewriting is set up carefully, the subgoals will blow up exponentially. It is sometimes convenient to unfold definitions, allowing the simplifier to prove some subgoals automatically. However, if we unfold everything, the remaining subgoals will be unreadable.

Lemmas of a peculiar form must be proved by induction. Fortunately, the necessary techniques appear to be similar for all protocols. We proved the secrecy of the session keys used in the digital envelopes and of nonces such as PANSecret. Secrecy of the PAN involves two theorems, depending upon whether the purchase request is signed or unsigned. We did not have to introduce new relations, as we did for cardholder registration.

A concern that emerged from our proof efforts is that many guarantees for cardholders and payment gateways depend upon the assumption that merchants are not compromised. This is due, not to a lack of effort on our side, but to an apparently unrelated feature of the design: the payment gateway is chosen by the merchant alone during the SET initiation process. His public and private keys are often used but his name is not confirmed (for example in a digital signature) either by the cardholder or by the merchant at any later stage. Thus, the payment gateway cannot be certain that the cardholder intended him to take part in the transaction. Message 3 involves six copies of the field XID (transaction identifier) and nine copies of the field PurchAmt (purchase amount), but it never mentions the identity of the intended payment gateway.

If the merchant is dishonest, there cannot be any guarantee that the right payment gateway has been selected. Furthermore, it is impossible to prove the agreement between the name and key of the payment gateway used by the cardholder and the name and key of the payment gateway authorizing the transaction. Although the failure of this property is disappointing, it does not appear to allow a significant attack within the protocol. It could only be exploited by a rogue payment gateway, who could induce an anomalous execution on another gateway.

This might generate unexpected behaviour. For instance, a rogue merchant could collude with a rogue payment gateway to harvest credit card numbers. The rogue payment gateway would then use a good payment gateway to authorize the transaction to disguise its misdeeds. This “attack” looks unrealistic, since a payment gateway is certified as such by a credit card institution. Cardholders have to trust it, just as they trust the credit card company itself. Adding the name of the payment gateway to the messages signed by the customer improves security only slightly.

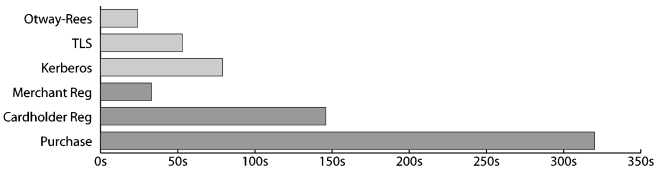
Thus, we reject the dualistic view that every protocol is either correct or vulnerable to attack. Anomalous executions that do little harm within the protocol cannot be called attacks. It is only after considering the environment in which the protocol operates and the trust relationships between the real embodiment of the protocol agents that anomalous executions could be classified as flaws of either protocol or documentation.

Digital envelopes complicate the statements of many guarantees. Agreement among principals obviously refers to important fields such as the order description and purchase amount. While we certainly hope the two parties will agree on which session key was used in a digital envelope, that property does not seem to be essential. We decided not to prove agreement on session keys because the value of this result did not justify the effort needed to prove it. Loosely speaking, we have proved that the keys on all locks (symmetric keys in digital envelopes) remained secret, that the contents of the luggage remained secret and unchanged, and that the sender was authentic. We have not proved that the luggage arrived with the same locks with which it was packed.

## 7 Conclusions

Our study demonstrates that enormous protocols such as SET are amenable to formal analysis. Such work is challenging, however. Understanding the documentation and defining a formal model can take months. Unfortunately, we did not record how much time we devoted to the various tasks and have to rely on memory. Piero Tramontano, a M.Eng. student at that time, devoted about 7 man-months to understanding SET under our supervision. While he concentrated on cardholder registration, much of this time was devoted to understanding the fundamentals of SET in general [10]. Completing the cardholder registration proofs required the dependency relation described in Sect. 5.2 and took perhaps 2 weeks. Merchant registration is simpler than cardholder registration and may have needed 2 weeks for its modelling and verification. For the purchase phase, we may have devoted 11 weeks. These numbers are very approximate. Modelling took longer than proof: the unsatisfactory status of some proved results sometimes led to changes in the model or lengthy examination of the SET documentation.

The proofs are still difficult. Isabelle may present the user with subgoals that are hundreds of lines long. Diagnosing a failed proof requires meticulous examination of huge and unintuitive formulae, where all abbreviations have been fully expanded. Such monstrosities impose a heavy burden on the computer, too. A simplification step can take 10 or 20 s on a 1.8-GHz processor. The bar chart shows the runtime required to execute the proofs for several protocols on a 1.8-GHz machine. There are three SET protocols (dark shading) and three others (light shading). These data are suggestive rather than compelling because minor changes to a proof script can cause major changes to the runtime. It suggests that merchant registration is very simple. Cardholder registration requires more effort, partly because it is longer and partly because it demands more secrecy proofs. The purchase protocol is by far the most difficult one.




It is not clear whether model checking could cope with this protocol's complexity. Specialized verification tools are more powerful than Isabelle, but they are less flexible. Even using Isabelle, the burden on the human verifier is too high to be increased further.

The single greatest advance would be a method of abstraction allowing constructions such as the digital envelope to be verified independently. We could then model these constructions abstractly in protocol specifications. In the case of SET, we could replace all digital envelopes by their abstract version. Assertions would become more concise; proofs would become much simpler. Unfortunately, abstraction in the context of security is ill understood and can mask grave flaws [39].

The other advance depends on protocol designers: they should provide a formal protocol definition worthy of the name. It should precisely specify several things:

1. A version of the message flow comprising the security features only,
2. A clear separation of features necessary to patch real-word cryptography (such as salt, which thwarts dictionary attacks) from abstract primitives (such as perfect hashing, encryption and signature),
3. The protocol's precise objectives, expressed as operational guarantees to each party,
4. The protocol's operating environment, including the threat model.

We are not saying that formal verification should be used during the design (though it might be desirable eventually), nor that the formal protocol definition should employ a logical formalism (designers would disagree on which one to use). We merely insist that the protocol documentation should clearly specify the items mentioned above. The implementers and maintenance staff would also benefit from a clear and precise specification. At present, we are forced to reverse engineer the protocol's core design from its documentation, and we have to guess what the protocol is supposed to achieve.

*Acknowledgements.* Comments from the anonymous reviewers were invaluable in giving the right security perspective to many issues discussed in the paper. The EPSRC grant GR/R01156/R01 *Verifying Electronic Commerce Protocols* supported Bella and Paulson. The MIUR-FIRB grants RBNE0195K5 and RBAU01P5SS and a former CNR fellowship supported Massacci.

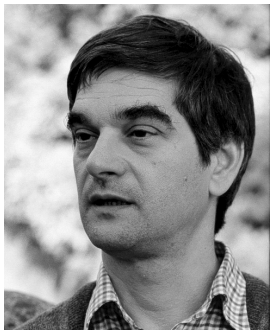
## References

1. Abadi M, Blanchet B (2003) Computer-assisted verification of a protocol for certified email. In: Cousot R (ed) Proc. 10th international symposium on static analysis (SAS'03), Lecture notes in computer science, vol 2694. Springer, Berlin Heidelberg New York, pp 316–335

2. Anderson R (1994) Why cryptosystems fail. Commun ACM 37(11):32–40
3. Armando A, Basin D, Bouallagui M, Chevalier Y, Compagna L, Mödersheim S, Rusinowitch M, Turuani M, Viganò L, Vigneron L (2002) The AVISS Security Protocol Analysis Tool. In: Proc. international conference on computer-aided verification (CAV-02). Lecture notes in computer science, vol 2404. Springer, Berlin Heidelberg New York
4. Armando A, Compagna L, Ganty P (2003) SAT-based model-checking of security protocols. In: Proc. 12th international FME symposium (FME03), Pisa, Italy, 8–14 September 2003
5. Asokan N, Schunter M, Waidner M (1997) Optimistic protocols for fair exchange. In: Proc. 4th ACM conference on computer and communication security (CCS-97). ACM Press/Addison-Wesley, Reading, MA, pp 7–17
6. Ateniese G, Steiner M, Tsudik G (1998) Authenticated group key agreement and friends. In: Proc. 5th ACM conference on computer and communication security. ACM Press, New York, pp 17–26
7. Basin D (1999) Lazy infinite-state analysis of security protocols. In: Baumgart R (ed) Secure networking – CQRE (Secure)'99, Lecture notes in computer science, vol 1740. Springer, Berlin Heidelberg New York, pp 30–42
8. Basin D, Denker G (2001) Maude versus Haskell: an experimental comparison in security protocol analysis. In: Futatsugi K (ed) Electronic notes in theoretical computer science, vol 36. Elsevier (North-Holland), Amsterdam
9. Bella G (2003) Inductive verification of smart card protocols. J Comput Secur 11(1):87–132
10. Bella G, Massacci F, Paulson L, Tramontano P (2000) Making sense of specifications: the formalization of set (extended abstract). In: Christianson B, Crispo B, Malcolm JA, Roe M (eds) Proceedings of the 2000 workshop on security protocols. Lecture notes in computer science, vol 2133. Springer, Berlin Heidelberg New York, pp 74–81
11. Bella G, Massacci F, Paulson LC (2002) The verification of an industrial payment protocol: the SET purchase phase. In: Atluri V (ed) Proc. 9th ACM conference on computer and communications security. ACM Press, New York, pp 12–20
12. Bella G, Massacci F, Paulson LC (2003) Verifying the SET registration protocols. IEEE J Select Areas Commun 21(1):77–87
13. Bella G, Massacci F, Paulson LC, Tramontano P (2000) Formal verification of cardholder registration in SET. In: Cuppens F, Deswarte Y, Gollman D, Waidner M (eds) Computer Security – ESORICS 2000. Lecture notes in computer science, vol 1895. Springer, Berlin Heidelberg New York, pp 159–174
14. Bella G, Paulson LC (1998) Kerberos version IV: inductive analysis of the secrecy goals. In: Quisquater et al. [37], pp 361–375
15. Blanchet B (1998) An efficient cryptographic protocol verifier based on prolog rules. In: Proc. 14th IEEE workshop on computer security foundations. IEEE Press, New York
16. Brackin S (1999) Automatically detecting authentication limitations in commercial security protocols. In: Proc. 22nd national conference on information systems security, October 1999.
`http://www.arca.com/proj_papers/brackin/Nissc99.pdf`
17. Burrows M, Abadi M, Needham RM (1989) A logic of authentication. Proc R Soc Lond 426:233–271
18. Carlucci Aiello L, Massacci F (2001) Verifying security protocols as planning in logic programming. ACM Trans Comput Logic 2(4):542–580
19. Chevalier Y, Vigneron L (2002) Automated unbounded verification of security protocols. In: Proc. international conference on computer-aided verification (CAV-02). Lecture notes in computer science, vol 2404. Springer, Berlin Heidelberg New York
20. Cohen E (2000) TAPS: A first-order verifier for cryptographic protocols. In: Proc. 13th IEEE Computer Security Foundations Workshop. IEEE Press, New York, pp 144–158
21. Jacquemard F, Rusinowitch M, Vigneron L (2000) Compiling and verifying security protocols. In: Parigot M, Voronkov A (eds) Proceedings of LPAR 2000. Lecture notes in computer science, vol 1955. Springer, Berlin Heidelberg New York, pp 131–160
22. Kessler V, Neumann H (1998) A sound logic for analysing electronic commerce protocols. In: Quisquater et al. [37]
23. Lowe G (1996) Breaking and fixing the Needham–Schroeder public-key protocol using CSP and FDR. In: Margaria T, Steffen B (eds) Proc. 2nd international workshop on tools and algorithms for the construction and analysis of systems (TACAS '96). Lecture notes in computer science, vol 1055. Springer, Berlin Heidelberg New York, pp 147–166
24. Lowe G, Hui ML (2001) Fault-preserving simplifying transformations for security protocols. J Comput Secur 9:3–46
25. Mastercard & VISA (1997) SET Secure Electronic Transaction: External Interface Guide, May 1997.
`http://www.setco.org/set_specifications.html`
26. Mastercard & VISA (1997) SET Secure Electronic Transaction Specification: Business Description, May 1997.
`http://www.setco.org/set_specifications.html`
27. Mastercard & VISA (1997) SET Secure Electronic Transaction Specification: Formal Protocol Definition, May 1997.
`http://www.setco.org/set_specifications.html`
28. Mastercard & VISA (1997) SET Secure Electronic Transaction Specification: Programmer's Guide, May 1997.
`http://www.setco.org/set_specifications.html`
29. Meadows C (1999) Analysis of the Internet Key Exchange protocol using the NRL Protocol Analyzer. In: Proc. SSP-99. IEEE Press, New York, pp 216–231
30. Meadows C (2003) Formal methods for cryptographic protocol analysis: emerging issues and trends. IEEE J Select Areas Commun 21(1):44–54
31. Meadows C, Syverson P (1998) A formal specification of requirements for payment transactions in the SET protocol. In: Hirschfeld R (ed) Proc. Financial Cryptography '98, Lecture notes in computer science, vol 1465. Springer, Berlin Heidelberg New York
32. Nipkow T, Paulson LC, Wenzel M (2002) Isabelle/HOL: A proof assistant for higher-order logic. Lecture notes in computer science tutorial, vol 2283. Springer, Berlin Heidelberg New York
33. Paulson LC (1998) The inductive approach to verifying cryptographic protocols. J Comput Secur 6:85–128
34. Paulson LC (1999) Inductive analysis of the internet protocol TLS. ACM Trans Inf Syst Secur 2(3):332–351
35. Paulson LC (2001) Relations between secrets: two formal analyses of the Yahalom protocol. J Comput Secur 9(3):197–216
36. Pereira O, Quisquater J-J (2001) Security analysis of the cliques protocols suites: first results. In: Dupuy M, Paradinas P (eds) Proc. IFIP TC11 16th annual working conference on information security (IFIP/Sec'01), vol 193. Kluwer, Dordrecht
37. Quisquater J-J, Deswarte Y, Meadows C, Gollmann D (eds) (1998) Computer Security – ESORICS 98. Lecture notes in computer science, vol 1485. Springer, Berlin Heidelberg New York
38. Focardi R, Gorrieri R (1997) The compositional security checker: a tool for the verification of information flow security properties. IEEE Trans Softw Eng 23(9):550–571
39. Ryan P, Schneider S (1998) An attack on a recurive authentication protocol. A cautionary tale. Inf Process Lett 65(15):7–16
40. Ryan PYA, Schneider SA (2000) The modelling and analysis of security protocols: the CSP approach. Addison-Wesley, Reading, MA
41. Schneider S (1998) Verifying authentication protocols in CSP. IEEE Trans Softw Eng 24(9):741–758
42. Shmatikov V, Mitchell JC (2000) Analysis of a fair exchange protocol. In: Proc. symposium on network and distributed system security (NDSS-00)
43. Song D (1999) Athena: an automatic checker for security protocol analysis. In: Proc. 12th IEEE Computer Security Foundations Workshop. IEEE Press, New York
44. Stoller SD (2001) A bound on attacks on payment protocols. In: Proc. 16th annual IEEE symposium on logic in computer science (LICS), June 2001

**Giampaolo Bella** is assistant professor at the University of Catania (Italy), where he teaches Computer Security. His main research interests lie in the analysis of crucial security properties using formal methods. His experience includes mechanical analysis using proof assistants, and manual analysis using the constraint satisfaction framework. Recently, he has worked for the verification of industrial security protocols within an EPSRC project focused on SET. He was a research associate at the Technical University of Munich (Germany) in the year 2000, after he received his Ph.D. from the Cambridge University Computer Laboratory. His Ph.D. dissertation, entitled "Inductive Verification of Cryptographic Protocols", was nominated by the examiners for the Distinguished Dissertation Award of the British Computer Society.

**Fabio Massacci** is associate professor (abilitated as full professor) in Informatics at the Universita' di Trento since September 2001. Since 2003 is rector's deputy for ICT services. He was assistant prof. at the Univ. of Siena since Mar/99. PhD in CompEng in Jun/1998 and MEng. at the University of Roma on Nov/93. In 2001 he was awarded the "Intelligenza Artificiale" prize by the Italian Association of Artificial Intelligence (AI*IA) for young researchers in AI under 35. We has won 4 post-doc fellowship from the National Reasearch Council (CNR), of which two for working at IRIT (CNRS) in France. He has worked on Automated Reasoning since he started his PhD in 1994 and on Security Protocols Verification since 1996, He is currently working on formal methods for security requirements engineering and security for web services.



**Lawrence Paulson** was born in 1955, Philadelphia, Pennsylvania. He studied Mathematics at the California Institute of Technology and received a B.S. in 1977. He received a Ph.D. in Computer Science from Stanford University in 1982. He proceeded to a postdoctoral position at the University of Edinburgh, Scotland, working under Robin Milner and Michael Gordon. In 1983, he accepted the position of Assistant Director of Research at the University of Cambridge, England. He was promoted to Reader in 1998 and to Professor in 2002.

His research concerns interactive theorem proving and its applications. His early work on Edinburgh LCF was incorporated into the HOL and Nuprl proof tools. In 1986 he introduced Isabelle, theorem prover that can work in a number of different formalisms. He has applied Isabelle to numerous case studies in specification and verification. His work on computer security dates from 1996, when he introduced the inductive approach to verifying cryptographic protocols.

He is married with two children.