Control Flow Analysis Can Find New Flaws Too*

Chiara Bodei[1], Mikael Buchholtz[2], Pierpaolo Degano[1],
Flemming Nielson[2], Hanne Riis Nielson[2]

1 Dipartimento di Informatica, Università di Pisa, Via F. Buonarroti 2, I-56127 Pisa, Italy. – {chiara,degano}@di.unipi.it; 2 Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads bldg 321, DK-2800 Kongens Lyngby, Denmark – {mib,nielson,riis}@imm.dtu.dk

Abstract. A previous study [6] showed how control flow analysis can be applied to analyse key distribution protocols based on symmetric key cryptography. We have extended both the theoretical treatment and our fully automatic verifier to deal with protocols based on asymmetric cryptography. This paper reports on the application of our technique – exemplified on the Beller-Chang-Yacobi MSR protocol, which uses both symmetric and asymmetric cryptography – and show how we discover an undocumented flaw.

1 Introduction

Formal analysis of security protocols has recently received a lot of attention from many different communities. Our attention has been on using control flow analysis and the main points we see in favour of this static analysis approach are the following: (*i*) the analysis is fully automatic and always terminating (often in low polynomial time); (*ii*) the analysis is correct with relation to a formal operational semantics of the calculus; and (*iii*) a single analysis step suffices for a variety of properties: different inspections of an analysis result permit to check different security properties of a protocol, with no need of re-analysing it several times.

We demonstrated in [6] that our technique is strong enough to report the known flaws on a range of key distribution protocols and to guarantee the absence of flaws on their amended versions. Here, we report on an extension to deal with asymmetric cryptography. Our approach is sufficiently robust so that only small incremental additions are needed to analyse protocols that use this encryption scheme. As a matter of fact,

* Supported in part by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies, under the IST-2001-32072 project DEGAS; the Danish SNF-project LoST; and the Italian MIUR-project MEFISTO.

our analysis can be further extended with other features typically used in protocol design as demonstrated in [11].

We have experimented with this new feature of our automatic verifier by considering some asymmetric key protocols among which is the well-known Needham-Schroeder (public key) [21]. Here, we felt reassured about the applicability of our extensions since we rediscovered the known attack by Lowe [16] and validated the correctness of the amendment.

Other experiments were carried out on the Beller-Chang-Yacobi MSR protocols [4], which use a combination of symmetric and asymmetric key cryptography to attain authentication in wireless networks. These protocols have received quite some attention in the literature, e.g. [12, 25, 9, 10], and our automatic verifier was able to rediscover the known flaws. To our surprise, we also found a family of parallel session attacks that, to the best of our knowledge [8], has not previously been documented.

Section 2 of this paper reviews our analysis technique including the extensions to handle asymmetric cryptography while Section 3 reports, in detail, on the application of the technique to the MSR protocols and demonstrates how it discovers the new flaws.

2 The Technique

2.1 The LySa Calculus

The LySa calculus is based on the π-calculus and the Spi-calculus, but it differs from these essentially in two aspects. The first is the absence of channels: LySa assumes to have one global communication medium to which all processes have access. The second is that the tests associated with input and decryption are expressed using pattern matching.

LySa consists of names, terms, and processes. Names, $N \in \mathcal{N}$, are used to model symmetric keys, nonces, messages, etc. as well as asymmetric keys m^+ and m^-. If m^+ is used for encryption then only m^- can be used for decryption, and vice versa, catering both for asymmetric encryption and digital signatures *à la* RSA [24]:

$$N ::= n \mid m^+ \mid m^-$$

Terms, E, are used to model messages built from names, variables, and encryption under a symmetric or an asymmetric key:

$E ::=$	*terms*
N	name ($N \in \mathcal{N}$)
x	variable ($x \in \mathcal{X}$)
$\{E_1, \cdots, E_k\}_{E_0}$	symmetric key encryption
$\{\|E_1, \cdots, E_k\|\}_{E_0}$	asymmetric key encryption

Symmetric key encryptions are tuples of terms $E_1, \cdots, E_k$ encrypted under a term E_0 representing the symmetric key, while asymmetric encryptions are tuples encrypted under an asymmetric key E_0, which should be one of a key pair. Processes, P, are given as

P ::=	*processes*
$(\nu\, n)P$	name creation
$(\nu_{\pm}\ m)P$	key pair creation
$\langle E_1, \cdots, E_k\rangle. P$	output
$(E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k). P$	input (with matching)
$\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k\}_{E_0}\ \mathsf{in}\ P$	
	symmetric decryption (with matching)
$\mathsf{decrypt}\ E\ \mathsf{as}\ \{\!\vert E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k\vert\!\}_{E_0}\ \mathsf{in}\ P$	
	asymmetric decryption (with matching)
$P_1 \mid P_2$	parallel composition
$!\,P$	replication
$\mathsf{0}$	the terminated process

The operator $(\nu\, n)P$ creates a new name, n, to be used as a nonce, a message, or a symmetric key and restricts the scope of n to the process P. The operator $(\nu_{\pm}\ m)P$ instead creates *two* new names, m^+ and m^-, to be used as a key pair in P. The process $\langle E_1, \cdots, E_k\rangle. P$ sends the k-tuple $\langle E_1, \cdots, E_k\rangle$ on the network and then continues as the process P. Correspondingly, the process $(E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k). P$ receives a k-tuple $\langle E'_1, \cdots, E'_k\rangle$ thereby removing the k-tuple from the network. This input is combined with pattern matching of the first j terms in the message and only succeeds if $E_i = E'_i$ for all $i \in [1, j]$. On successful matching, the variables $x_{j+1}, \cdots, x_k$ are bound to the remaining $k - j$ terms $E'_{j+1}, \cdots, E'_k$. Syntactically, a semi-colon separates the components where matching is performed from those where binding takes place. The same simple form of patterns is also used for decryption — a more flexible choice of patterns is explored in [11].

The process $\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k\}_{E_0}\ \mathsf{in}\ P$ attempts to decrypt E with the symmetric key E_0. The decryption succeeds only if the term E is of the form $\{E'_1, \cdots, E'_k\}_{E'_0}$ and the key E'_0 is the same as E_0. Again pattern matching takes place so if furthermore $E_i = E'_i$ for all $i \in [1, j]$ then decryption succeeds and the process behaves as $P[E'_{j+1}/x_{j+1}, \ldots, E'_k/x_k]$. Similarly, the process $\mathsf{decrypt}\, E\ \mathsf{as}\ \{\!\vert E_1, \cdots, E_j;$ $x_{j+1}, \cdots, x_k\vert\!\}_{E_0}\ \mathsf{in}\, P$ attempts to decrypt E with the asymmetric key E_0 and succeeds provided that $E = \{\!\vert E'_1, \cdots, E'_k\vert\!\}_{E'_0}$ and that (E_0, E'_0) is a pair consisting of a public key and its private counterpart; it is irrelevant which is which, so catering for both asymmetric encryption and for digital signature validation.

Security Properties. Anticipating our analysis in the next section, we are interested in two security properties: *secrecy* and message *authentication*. Our analysis will explicitly represent the knowledge of the attacker and, thus, secrecy of any given value may easily be determined simply by inspecting whether the value is in the knowledge of the attacker or not.

Secondly, we consider a message authentication property that, informally, describes whether messages *end up* at the right places. The attacker can, of course, redirect any network messages and there will, in general, be no guarantee that they end up in the right places. Instead, we focus on the messages for which the attacker does not have full control, namely the ones where cryptography has been applied as some kind of safeguard.

To describe the message authentication intentions of protocols, we decorate their text with labels, called *crypto-points*, and with *assertions* specifying the intended origin and destination of encrypted messages. Crypto-points ℓ are from some enumerable set $\mathcal{C}$ (disjoint from $\mathcal{N}$ and $\mathcal{X}$) and are mechanically attached to program points where encryption and decryption occur. Crypto-points and assertions are added to the syntax and encryptions will, thus, have the form:

$$\{E_1, \cdots, E_k\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}] \qquad \text{or} \qquad \{\!|E_1, \cdots, E_k|\!\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}]$$

where ℓ is the crypto-point where the encryption takes place while $[\mathsf{dest}\ \mathcal{L}]$ is the assertion that specifies the intended crypto-points $\mathcal{L} \subseteq \mathcal{C}$ for decryption of the term. Similarly, a decryption process will be of the form:

$$\begin{array}{l}\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k\}^{\ell}_{E_0}\ [\mathsf{orig}\ \mathcal{L}]\ \mathsf{in}\ P\ \ \text{or}\\ \mathsf{decrypt}\ E\ \mathsf{as}\ \{\!|E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k|\!\}^{\ell}_{E_0}\ [\mathsf{orig}\ \mathcal{L}]\ \mathsf{in}\ P\end{array}$$

where ℓ is point of decryption while $[\mathsf{orig}\ \mathcal{L}]$ specifies the intended encryption points $\mathcal{L} \subseteq \mathcal{C}$ at which E was created.

Semantics. As far as the semantics is concerned, we consider two variants: (i) the first is the standard *reduction semantics* $P \rightarrow P'$ that ignores the annotations, while (ii) the second, called the reference monitor semantics $P \rightarrow_{\mathsf{RM}} P'$, checks annotations when performing the decryptions and *aborts* the execution if the conditions are violated. More specifically, decryptions should only occur at the crypto-points designated when the corresponding encryptions were made, and vice versa. For example, the check $(\ell \in \mathcal{L}' \wedge \ell' \in \mathcal{L})$ is necessary for symmetric decryption of the form

$$\mathsf{decrypt}\ \{E_1, \cdots, E_k\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}]\ \mathsf{as}\ \{E'_1, \cdots, E'_j;\ x_{j+1}, \cdots, x_k\}^{\ell'}_{E'_0}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P$$

and analogously for the asymmetric case. When the origin and destination are not the expected ones, we say that the reference monitor aborts the execution i.e. there is no next configuration.

2.2 Control Flow Analysis

The aim of the analysis is to safely approximate when the reference monitor may abort the execution of a process P. The analysis roughly computes an over-approximation of:

- the messages that may flow on the network, $\kappa \subseteq \wp(\mathcal{V}^*)$, and of
- the values to which a given variable may be bound $\rho : \mathcal{X} \rightarrow \wp(\mathcal{V})$

where $\mathcal{V}$ is the set of values (being terms with no free variables). For example, an analysis of the process

$$\langle A, B, \{K\}^{\ell_A}_{K_A}[\mathsf{dest}\ \ell_S]\rangle \mid (A; x, y).\mathsf{decrypt}\ y\ \mathsf{as}\ \{;\ z\}^{\ell_S}_{K_A}\ [\mathsf{orig}\ \ell_A]\ \mathsf{in}\ P$$

will tell that $\langle A, B, \{K\}^{\ell_A}_{K_A}[\mathsf{dest}\ \ell_S]\rangle \in \kappa$ and, since pattern matching in both input and decryption succeeds, that $B \in \rho(x)$, $\{K\}^{A}_{K_A}[\mathsf{dest}\ \ell_S] \in \rho(y)$, and $K \in \rho(z)$. The analysis result represents a conservative over-approximation of the values present in the execution of a process. Such values, e.g. the ones above, are guaranteed to be recorded in the analysis result but over-approximation may cause additional *false positives* to also appear. A "good" analysis is one that in practice gives few false positives and our analysis behaves well in this respect. For example, no false positives arose in the study presented here.

An additional *error component* ψ collects pairs of crypto-points (ℓ, ℓ'), where the assertions in annotations may be violated. Intuitively, $(\ell_A, \ell_B) \in \psi$ indicates that something encrypted at the crypto-point ℓ_A may *wrongly* be decrypted at the crypto-point ℓ_B. Conversely, if ψ is empty then the protocol is correct with respect to the assertions.

A triple (ρ, κ, ψ) is called *an estimate* when it correctly describes the behaviour of a process P. Formally, the analysis is specified as a Flow Logic with judgements $\rho, \kappa \models P : \psi$ that for each process P gives a formula that constraints the analysis components according to the behaviour of P. Our implementation of the analysis computes the least estimate that satisfies these judgements.

Our analysis is semantically correct regardless of the way the semantics is parameterised and there is a subject reduction result both for the standard and the reference monitor semantics: if $\rho, \kappa \models P : \psi$, then the

1. $B \rightarrow A : B, K_B^+$	1. $B \rightarrow A : B, \{\!\|B\|\!\}_{K_U^-}, K_B^+$
2. $A \rightarrow B : \{\!\|K\|\!\}_{K_B^+}$	2. $A \rightarrow B : \{\!\|K\|\!\}_{K_B^+}$
3. $A \rightarrow B : \{A, \{\!\|A\|\!\}_{K_U^-}\}_K$	3. $A \rightarrow B : \{A, \{\!\|A\|\!\}_{K_U^-}\}_K$

Fig. 1: MSR protocol (left) and Improved MSR protocol (right) [4].

same triple (ρ, κ, ψ) is a valid estimate for all the states passed through in an execution of P, i.e. for all the derivatives of P.

Additionally, when analysing a process P if the error component ψ is empty then the reference monitor *cannot abort* the execution of P. This means that our analysis correctly predicts when we can safely dispense with the reference monitor.

Hardest Attackers. Protocols are executed in an environment where there may be malicious attackers. We consider attackers in the style of Dolev and Yao: such attackers can receive everything flowing on the network, can decrypt every ciphertext if they know the key, can compose any message with the terms they know and send it. Furthermore, they will initially have access to their own names $n_\bullet, K_\bullet^+$, and $K_\bullet^-$; the last two forming an asymmetric key pair. We have a formula $\mathcal{F}_{DY}$ characterising all attackers having these features at analysis level i.e. using ρ and κ. The attacker keeps its knowledge in a special variable $z_\bullet$, in terms of the analysis, in $\rho(z_\bullet)$. The initial knowledge can be increased: for example, for each message $\langle V_1, \cdots, V_k \rangle$ included in κ, we have that $V_i \in \rho(z_\bullet)$ for all $i \in [1, k]$. This is the static counter-part of the fact that the attacker can receive every message flowing on the network. Moreover, the attacker has a special crypto-point called $\ell_\bullet$ and the formula can also check annotations. The formula $\mathcal{F}_{DY}$ is therefore such that whenever an estimate (ρ, κ, ψ) satisfies $\mathcal{F}_{DY}$ then $\rho, \kappa \models Q : \psi$ for all attackers Q. Actually, there exists a process – a "hardest attacker" [22] – which has all the capabilities of this formula.

Our analysis suffices to statically guarantee the message authentication property for a process P if it results in a empty error component i.e. if $\rho, \kappa \models P : \emptyset$ and $(\rho, \kappa, \emptyset)$ satisfies the formula $\mathcal{F}_{DY}$ describing the attacker. The implementation of the analysis, including the attacker, runs in low polynomial time in the size of the process P.

3 Analysis of the MSR Protocols

The MSR protocols by Beller, Chang, and Yacobi [4] are intended for authentication between portables, denoted A, and base stations, denoted

B, that communicate through wireless networks. In [4], the protocols are described using a Modular Square Root algorithm for asymmetric cryptography having the advantage that different principals need not use the same amount of computational power to execute the protocol. In the protocol narrations in Figure 1 we have abstracted these operations to standard cryptographic primitives, writing $\{m\}_K$ for symmetric key encryption of m under K, $\{\!|m|\!\}_{K^+}$ for asymmetric key encryption of m under K^+, and $\{\!|m|\!\}_{K^-}$ for m signed under K^-.

The MSR protocol relies on an off-line certificate authority U that issues signed certificates of the identities (the name A and B) of portables and base stations. When a portable A arrives at a new base station B it first sends a dummy message on the network to alert the base station that it wants to authenticate. The base station replies with message 1 in the protocol narration on Figure 1 thereby sending its public key. The portable generates a fresh session key K and sends its back to the base station along with its certificate in messages 2 and 3. The MSR protocol and the Improved MSR protocol differ only in the certificate in the first message intended to give extra authentication capabilities as detailed below.

3.1 Formalising the Protocols in LySa

The MSR protocols shown in Figure 1 are straightforward to encode in LySa. Each protocol is encoded in a scenario where there is

- m *portables*, A_i for $i = 1 \ldots m$,
- n *base stations*, B_j for $j = 1 \ldots n$, and
- one *certificate authority*, U.

This scenario corresponds to the (implicit) assumption of [4] that no base station will play the role of a portable or vice versa, and that the certificate authority is a separate principal. Each base station, B_j, is equipped with a key pair (K_j^+, K_j^-) where the second component will be kept secret. Furthermore, the certificate authority has a signature key pair (K_U^+, K_U^-) and initially all A_i's and B_j's know the signature validation key K_U^+ along with their own certificates signed with K_U^-.

We encode the protocol such that each portable A_i simultaneously and repeatedly is willing to communicate with each base station B_j and vice versa. This corresponds to a setup where the portables are within the coverage of several base stations at the same time, which is quite realistic in a wireless network where the coverage of base stations overlap. Finally, we encode all messages such that source and destination

```
1   $(\nu_{\pm}\ K_U)\,\langle {K_U}^{+}\rangle\ \mid$
2   $\mid_{i=1}^{m}\ \mid_{j=1}^{n}\ !(B_j, A_i, B_j;\ xC_{ij}, xK_{ij}).$
3           decrypt $xC_{ij}$ as $\{\!|B_j;\ |\!\}^{a1_i}_{K_U{+}}$ [orig $b1_j$] in
4           $(\nu\, K_{ij})\langle A_i, B_j, \{\!|K_{ij}|\!\}^{a2_i}_{xK_{ij}}$ [dest $b2_j$]$\rangle.$
5                   $\langle A_i, B_j, \{A_i, \{\!|A_i|\!\}^{a3_i}_{K_U{-}}\}^{a4_i}_{K_{ij}}$ [dest $b3_j$]$\rangle.\,0$
6   $\mid (\nu_{\pm}{}_{j=1}^{n} K_j)$
7   $\mid_{j=1}^{n}\ \mid_{i=1}^{m}\ !\langle B_j, A_i, B_j, \{\!|B_j|\!\}^{b1_j}_{K_U{-}}, K_j^{+}\rangle.$
8           $(A_i, B_j;\ y1_{ij}).$
9           decrypt $y1_{ij}$ as $\{\!|;\ y2_{ij}|\!\}^{b2_j}_{K_j^{-}}$ in
10          $(A_i, B_j;\ y3_{ij}).$
11          decrypt $y3_{ij}$ as $\{A_i;\ y4_{ij}\}^{b3_j}_{y2_{ij}}$ [orig $a4_i$] in
12          decrypt $y4_{ij}$ as $\{\!|A_i;\ |\!\}^{b4_j}_{K_U{+}}$ [orig $a3_i$] in 0
```

Fig. 2: The Improved MSR protocol in LySa.

addresses are explicitly added as the first two components i.e. in the format $\langle \mathit{source}, \mathit{destination}, \mathit{message_1}, \cdots, \mathit{message_k}\rangle$. This ensures that all addresses are sent in clear on the network and, therefore, that the security of the protocol will not depend on keeping the addresses secret.

The LySa encoding of the Improved MSR protocol is shown in Figure 2. In line 1 the keys of the certificate authority are distributed to the principals, line 2–5 describe the part of m portables while line 6–12 describe the part of the n base stations.

Each point of encryption and decryption in the process is annotated with a label ℓ_i or ℓ_j for portables and base stations, respectively, and ℓ is chosen such that each crypto-point is unique. Furthermore, annotations of the intended destination and origin of encrypted messages are included. For example, in line 4, where a portable A_i is interested in talking to a base station B_j, the annotation [dest $b2_j$] says that the encrypted message is intended to be decrypted by a base station B_j at its second crypto-point, only. This corresponds to what is intended in the informal protocol narration in Figure 1 when writing "$A \to B\ :\ \{\!|K|\!\}_{K_B^{+}}$". Similarly in line 5, the second message encrypted by the portable is intended to reach B_j's fourth crypto-point, only, and so on.

Correspondingly, annotations are added at decryption points as in line 3, where [orig $b1_j$] specifies that the certificate validated here is supposed to be the one from B_j. At the first decryption made by the base station in line 9 there is no check of origin of the message, since the base station does not expect that only particular principals may use its public

$i = 1 \ldots m$, $j = 1 \ldots n$	Error component ψ	Attacker's basic knowledge $\rho(z_\bullet) \cap \mathcal{N}$
MSR	$(a2_i, \ell_\bullet)$, $(a4_i, \ell_\bullet)$, $(\ell_\bullet, b3_j)$, $(a2_i, b2_j)$, $(a4_i, b3_j)$	$n_\bullet, K_\bullet^+, K_\bullet^-, A_i, B_j, K_{ij}, K_j^+, K_U{}^+$
Improved MSR	$(a2_i, \ell_\bullet)$, $(a4_i, \ell_\bullet)$, $(\ell_\bullet, b3_j)$, $(a2_i, b2_j)$, $(a4_i, b3_j)$	$n_\bullet, K_\bullet^+, K_\bullet^-, A_i, B_j, K_{ij}, K_j^+, K_U{}^+$

Fig. 3: The result of running the analysis.

key. When the base station gets further in the execution of a protocol run it will, however, expect that only messages from A_i may be successfully decrypted as specified in line 11 and 12.

3.2 The Analysis Result

The control flow analysis of the MSR protocols gives the results reported in Figure 3. At a first glance it is clear that the protocols are flawed both because there is a non-empty error component, ψ, and because the attacker may learn all the session keys K_{ij}. That is, there is both a breach of *authentication* as specified in the annotations of the protocol and a breach of *secrecy* of the session keys.

The MSR protocol itself is meant to provide the base stations with authentication of the portables as well as to provide secrecy of the session keys. The Improved MSR protocol contains an extra certificate in the first message in order to provide additional authentication of the base stations to the portables i.e. to provide mutual authentication. Below we use the analysis result to evaluate the extent to which these goals are met.

Inspecting the attacker's knowledge in the analysis result in Figure 3, we see that neither of the protocols provide session key secrecy and this corresponds to what has previously been reported by Carlsen [12]. He also describes an attack on the portable's authentication of base station as depicted in the attack sequence on Figure 4 (left) where, at the end of the protocol run, the portable A_1 thinks it is talking to B_1 but is really talking to the attacker M whose public key we denote $K_\bullet^+$.

We may compare the attack sequence on Figure 4 (left) to the elements in the error component ψ in Figure 3. In the last two messages of the attack sequence something encrypted by the portable A_1 is decrypted by the attacker as reported by the two families of elements $(a2_i, \ell_\bullet)$ and $(a4_i, \ell_\bullet)$ in the error component.

Note that on decryption of message 2.2 in the attack sequence on Figure 4 (left) the attacker learns session key K_{11}. This gives the attacker

1.1. $B_1 \rightarrow M(A_2) : B_1, \{\!\vert B_1\vert\!\}_{K_U^-}, K_B^+$	1.1 $B_1 \rightarrow M(A_1) : B_1, \{\!\vert B_1\vert\!\}_{K_U^-}, K_{B_1}^+$
2.1. $M(B_1) \rightarrow A_1 : B_1, \{\!\vert B_1\vert\!\}_{K_U^-}, K_\bullet^+$	2.1 $B_2 \rightarrow M(A_1) : B_2, \{\!\vert B_2\vert\!\}_{K_U^-}, K_{B_2}^+$
2.2. $A_1 \rightarrow M(B_1) : \{\!\vert K_{11}\vert\!\}_{K_\bullet^+}$	2.1′ $M(B_2) \rightarrow A_1 : B_2, \{\!\vert B_2\vert\!\}_{K_U^-}, K_{B_1}^+$
2.3. $A_1 \rightarrow M(B_1) : \{A_1, \{\!\vert A_1\vert\!\}_{K_U^-}\}_{K_{11}}$	2.2 $A_1 \rightarrow M(B_2) : \{\!\vert K_{12}\vert\!\}_{K_{B_1}^+}$
	2.3 $A_1 \rightarrow M(B_2) : \{A, \{\!\vert A\vert\!\}_{K_U^-}\}_{K_{12}}$
	1.2′ $M(A_1) \rightarrow B_1 : \{\!\vert K_{12}\vert\!\}_{K_{B_1}^+}$
	1.3′ $M(A_1) \rightarrow B_1 : \{A, \{\!\vert A\vert\!\}_{K_U^-}\}_{K_{12}}$

Fig. 4: Two kinds of attacks on Improved MSR: key spoofing attacks [12] (left) and new parallel session attacks (right).

the ability to create messages of the form of the protocol's third message giving a breach of the base station's authentication of the portable. Again this attack was reported by Carlsen [12] and is also evident in our analysis result as the element $(\ell_\bullet, b3_j)$ in ψ, saying that something encrypted at the attacker may wrongfully be decrypted at B_j.

However, the remaining two elements in ψ, $(a2_i, b2_j)$ and $(a4_i, b3_j)$, cannot be explained by any of the attacks previously reported in the literature. Indeed they represent a previously undocumented class of parallel session attacks in the style of the attack sequence on the right of Figure 4. At the end of the protocol run the portable A_1 thinks it is talking to B_2 but is really talking to B_1 explaining the unintended cross-over between information constructed by legitimate principals leading to errors, $(a2_i, b2_j)$ and $(a4_i, b3_j)$, as reported by our control flow analysis.

The certificate in the first message of Improved MSR has very little effect and indeed our analysis results in Figure 3 report the same errors for both components. The attack sequences in Figure 4 also work on the simpler MSR protocol (provided that the certificates are removed, of course) though, strictly speaking, they are not attacks on MSR since that protocol is not intended to provide authentication of the base stations.

4 Conclusion

This paper advances our application of static analysis technology to the analysis of key distribution protocols by incorporating also perfect asymmetric cryptography. As expected we are able to find the well-known flaws of protocols and to validate the corrected versions. When considering the Improved MSR protocol by Beller, Chang and Yacobi, we were able to pinpoint a new class of attacks that to the best of our knowledge [8] have not been reported previously.

When assessing our method it is important to stress that it is a fundamentally different approach from at least some of its competitors, e.g. model checking and theorem proving [16, 19, 20, 18, 13, 3, 17, 7, 23]. Protocol analysis really amounts to capturing the dynamic behaviour of a communication protocol; however, this may be an undecidable property due to the richness of the term language.

In our view, model checking approaches the problem by trying to characterise an under-approximation of this recursively enumerable set. If there is a flaw, model checking may be able to pinpoint it, whereas it cannot in general guarantee the absence of flaws. This is very clear e.g. in the works of [3] although one may impose boundedness conditions to try to achieve termination (but with weaker guarantees).

Our approach on the other hand, tackles the problem by trying to characterise an over-approximation of the recursively enumerable set. Hence if we report no flaws indeed there will be no flaws; also our methods are guaranteed always to terminate. The potential weakness of our approach is that, because of the over-approximation, it may report ”flaws” that really are not there. In our work so far, this has only happened rarely and no such phenomena arose in the work reported in the present paper. As far as the computational complexity is concerned we operate in low polynomial time in the size of the expanded LySa process encoding the protocol. Similar remarks could be said about other approximation based techniques [5, 14, 2] though [5, 14] have significantly higher complexity, while [2] focuses on secrecy only.

Expanding our comparison to other approaches we would describe theorem proving much in the same vein as model checking. Here one is searching for potential proofs of correctness and the risk is that one misses the proof actually showing correctness and that the theorem proving then does not terminate. Type systems, on the other hand are somewhat closer to our approach, especially for those type systems that have principal types; however, many type systems used for protocol analysis [1, 15] would seem to admit polynomial time type checking but no principal types, and therefore would seem to require exponential time as far as practical systems are concerned.

References

1. M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 5(46):18–36, 1999.
2. R. Amadio and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *Proc. of CONCUR'02*, LNCS 2421, pages 499–514. Springer, 2002.

3. A. Armando, D. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The AVISS security protocol analysis tool. In *CAV'02*, LNCS 2404, pages 349–353. Springer, 2002.
4. M. J. Beller, L.-F. Chang, and Y. Yacobi. Privacy and authentication on a portable communications system. *IEEE Journal of Selected Areas in Communications*, 11(6):821–829, 1993.
5. B. Blanchet. From secrecy to authenticity in security protocols. In *Proc. of SAS'02*, LNCS 2477, pages 342–359. Springer, 2002.
6. C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. Riis Nielson. Automatic validation of protocol narration. In *Proc. of CSFW'03*, pages 126–140. IEEE, 2003.
7. D. Bolignano. An approach to the formal verification of cryptographic protocols. In *Proc. of 3rd ACM Conf. on Computer and Communications Security*, pages 106–118. ACM Press, 1996.
8. C. Boyd, 2003. Personal e-mail communication.
9. C. Boyd and A. Mathuria. Key establishment protocols for secure mobile communication: a critical survey. *Computer Communications*, 23:575–587, 2000.
10. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.
11. M. Buchholtz, F. Nielson, and H. Riis Nielson. A calculus for control flow analysis of security protocols. *International Journal of Information Security*, To appear.
12. U. Carlsen. Optimal Privacy and Authentication on a Portable Communications System. *Operating Systems Review*, 28(3):16–23, 1994.
13. E.M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with Brutus. *ACM Trans. on S/W Engineering and Methodology*, 9(4):443–487, 2000.
14. H. Comon, V. Cortier, and J. Mitchell. Tree automata with memory, set constraints and ping-pong protocols. In *Proc. of ICALP'01*, LNCS 2305. Springer., 2001.
15. A. D. Gordon and A. Jeffrey. Types and Effects for Asymmetric Cryptographic Protocols. In *Proc. of CSFW'02*, pages 77 –91, 2002.
16. G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.
17. C. Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.
18. J. Millen. CAPSL web site. http://www.csl.sri.com/users/millen/capsl/.
19. J. Millen. The Interrogator: A tool for cryptographic protocol security. In *Proc. of Symposium on Security and Privacy*, pages 134–141. IEEE, 1984.
20. J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murϕ. In *Proc. of S&P*, pages 141–153. IEEE, 1997.
21. R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.
22. F. Nielson, H. Riis Nielson, and R. R. Hansen. Validating firewalls using flow logics. *Theoretical Computer Science*, 283(2):381–418, 2002.
23. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.
24. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosytems. *Communications of the ACM*, 21(2):120–126, 1978.
25. Y. Zheng. An authentication and security protocol for mobile computing. In *Proc. of 1996 World Conference on Mobile Communications*, pages 249–57. Chapman & Hall, 1996.