An Analysis of Integrity Services in Protocols*

Kapali Viswanathan, Colin Boyd, and Ed Dawson

Information Security Research Centre, Queensland University of Technology,
GPO Box 2434, Brisbane, Q 4001, Australia,
kapali@isrc.qut.edu.au, {boyd,dawson}@fit.qut.edu.au

Abstract. An analysis of integrity services in cryptologic protocols is presented. The informal syntax, to be presented, attempts to model the integrity service as a property that is transferred from a key to a message. The message can, in turn, be a key. The modeling presupposes confidentiality and integrity to be the atomic properties or services offered by cryptologic algorithms. More complex algorithms and protocols, such as those for digital signature, identification protocols and non-malleable encryption, are considered to be ensembles of these services. This paper concentrates only on the analysis of the integrity service in signature techniques based on the proof of knowledge of discrete logarithm. The paper will demonstrate the usefulness of this modeling by identifying flaws in the recent proposals for an efficient electronic cash system and a key-recovery system.

Keywords : Confidentiality, integrity, representation of cryptologic goals.

1 Introduction

Confidentiality and integrity services are the atomic properties that are required for the construction of cryptologic protocols. The work on network security architectures by Rueppel [14] is an example for a research with a similar view. These properties can be viewed as follows: keys provide service (confidentiality or integrity) to messages. The importance of entities (like Alice or Bob) is deliberately avoided in subsequent definitions and analyses in order to facilitate a *key-centric view* of cryptosystems[1], which may be more appropriate for the representation, analysis and design of cryptosystems. Such an approach does not require any form of protocol idealisation [5], which may create more difficulties in the analysis of protocols. Moreover, since the idealisation functions do not have an inverse mapping (de-idealisation functions), the analysis techniques employing such functions may not be useful directly in the design of protocols.

A cryptosystem can be viewed to be a composition of integrity and confidentiality services, which can be considered to be independent of each other. Although integrity and confidentiality services are not totally independent, the

* Research Supported by the Australian Research Council grant A49804059

[1] This is as opposed to an *entity-centric view*, such as that of the BAN logic [5].

C. Pandu Rangan, C. Ding (Eds.): INDOCRYPT 2001, LNCS 2247, pp. 175–187, 2001.

results of this paper will be logically consistent. This is because if the relationship between the two services were to be represented syntactically, the syntax will only add more functionalities to the model and will not remove any.

Due to this view, cryptosystems may be decomposed into an integrity component and a confidentiality component. This decomposition when represented in a suitable fashion will result in a simple characterisation of the goals of the cryptosystem – that is the integrity goal and the confidentiality goal. Many proposals in recent times, knowingly or unknowingly, have neglected the integrity goal of the cryptosystem. The negligence often results in deficient cryptosystems, which may be highly undesirable for many applications.

The concern of this paper is an informal technique for the representation of the integrity goal. There exists many papers that have attempted to represent the confidentiality goal, such as the paper by Abadi and Rogaway [1]. So, this paper will not deal with the representation of the confidentiality service. Section 2 presents an analysis of the integrity goal. The subsequent sections will employ the proposed technique to analyse the the electronic cash system proposed by Radu, Govaerts and Vandewalle [12], and the fraud detectable key recovery scheme by Verheul and van Tilborg [17].

2 An Integrity Verification Technique

The informal working definitions for the integrity and confidentiality services are as follows:

Definition 1 Confidentiality *is the service that grants* access *to the message corresponding to the cipher-text when the* access *to the key is available.*

Definition 2 Integrity *is the service that determines the* immutability *of a message corresponding to a cipher-text when the* immutability *of the key has been determined.*

These definitions express succinctly the importance of the confidentiality and integrity properties of the keys in cryptosystems. The aim of any cryptosystem is to *maintain* the confidentiality and integrity properties of the messages with respect to the corresponding properties of the keys.

The transfer of a cryptologic property from a key to a message will be represented as follows:

$$K \xrightarrow{SERVICE,C} M$$

where, $SERVICE \in \{\mathcal{C}, \mathcal{I}\}$ is the type of service, $\mathcal{C}$ is the keyword for the confidentiality service and $\mathcal{I}$ is the keyword for the integrity service. Confidentiality is the private view of the participants and integrity is the public view. The terms private and public are relative depending upon the assumptions about the ownership of various keys. Since, this paper is interested only in the characterisation of the integrity service, the subsequent representations will present only the graphs for the transfer of the integrity service.

While characterising the integrity goal of any system (such as the Schnorr signature scheme, Brands' e-cash scheme), the model will account only for the verification equations. This abstraction is essential to model the *unpredictable behaviour* of the signer. The signer's behaviour is unpredictable because the verifier does not necessarily trust the signer. The behaviour of the verifier is not modeled because it is assumed that the verifiers perform the verifications to safe-guard their interests. Moreover, it is not the concern of cryptology to force the verifier to act properly during and after the verification process.

This section presents a protocol developer's view, as opposed to a cryptologic algorithm developer's view, of the general purpose signature schemes and an informal syntax for the representation of the transfer of service from keys to messages. The results are then extended to represent the Schnorr signature scheme [15] in Section 2.1. Section 2.2 contains a discussion on Schnorr-type blind signature schemes [7,4] and outlines the subtleties that protocol designers must be aware of.

2.1 Characterising Signature Schemes

The following representation for the signature schemes will be employed in this paper:

$$\langle PublicKey \rangle \xrightarrow{\langle Ciphertexts \rangle} \langle Message \rangle$$

The term $\langle Ciphertexts \rangle$ includes the result of any cryptographic operation, such as encryption and signature operations. For example, if $y = g^x \bmod p$ for a suitable value of p and g, then y is a cipher-text. There may be one or more individual cipher-texts in the system. Usually, the signature process is computationally expensive and the messages are arbitrarily long. Additionally, the use of secure hash functions improve the security of the verification equations. Therefore, suitable message digest (symmetric key) techniques are employed. This gives raise to two techniques.

The first technique is to sign the message digest. Suppose that an RSA public-key pair [13], $[e, n]$, is employed to sign a message, m, employing a secure hash function, $\mathcal{H}$, to generate the following verification equations:

$$c \stackrel{?}{=} \mathcal{H}(m, A)$$
$$r \stackrel{?}{=} c^e \bmod n$$

then $[c, r]$ are the signature tuples. This technique is represented as follows:

$$\Big((A \xrightarrow{c} m) \wedge ([e, n] \xrightarrow{r} c) \Big)$$

where:

1. c is the message digest;

2. A is the symmetric key. When an unkeyed hash function is employed, $A = \emptyset$, which is the *null key*.
3. m is the message to be signed.
4. $[e, n]$ is the public key of the signer.
5. r is the signature cipher-text.

Henceforth, the logical "and" operation will be represented by the symbol $\wedge$. This operator suggests that individual verification equations must output `true` for the verification system to output `true`. Note that the $\emptyset$ key represents the no key scenario and is known globally to all participants. Also note the myriad of protocol design possibilities when *SymmetricKey* is not equal to the $\emptyset$ key.

The second technique is to sign a symmetric key that would provide integrity service to the message. The technique proposed by Fiat and Shamir [9], and adopted by Schnorr [15] is a good example. Such a signature technique is represented as follows:

$$(\langle PublicKey\rangle \xrightarrow{\langle SignatureCiphertext\rangle} \langle SymmetricKey\rangle \xrightarrow{\langle MessageDigest\rangle} \langle Message\rangle)$$

The symmetric key, in this case, cannot be $\emptyset$ (null key). Note that the representation, by itself, does not suggest that the signature cipher-text provides non-repudiation service to the message, rather it suggests the integrity service for the symmetric key, which in turn provides integrity service to the message. This is because the representation deals with a lower level view to trace the flow of integrity service, which is more basic than the non-repudiation service. In order to achieve the non-repudiation service for the message, a one-to-one relationship between the symmetric key and the message, which in the Schnorr signature scheme is achieved by a one-to-one relationship between the signature cipher-text and the message digest, is essential. The rest of this section will explain this form of representation in detail.

A tuple $[r,\ c]$ is a valid Schnorr signature on a set of messages m by the public key $[g, y, p]$ (henceforth the symbol p, representing the prime number, will be omitted whenever it can be implicitly understood), if the following equation holds:

$$c \stackrel{?}{=} \mathcal{H}(m, A)$$

where, $\mathcal{H}$ is a secure hash function, c is the message digest and $A = y^c g^r$ is the symmetric key. The integrity goal of the Schnorr signature scheme can be expressed as follows:

$$\left([g, y] \stackrel{[c,r]}{\rightarrow} A \stackrel{c}{\rightarrow} m\right) \tag{1}$$

That is a *trusted* public key, $[g, y]$, provides integrity service to a symmetric key, A, by employing the cipher-texts, $[c, r]$. The symmetric key, A, in turn provides integrity service to the message, m, by employing the cipher-text c. The same value of the cipher-text, c is employed by the public key and the symmetric key.

It is important to note that in Schnorr-type signature schemes, the structure of A with respect to g, is similar to the structure of y with respect to g. That is by knowing the discrete logarithm, $\log_g y$ and the signature tuple, it is possible to know the value of $\log_g A$, and vice versa. This is an important requirement to prevent the generation of multiple signature transcripts from a single Schnorr signature. Henceforth, the $\big(\big)$ delimiter will separate verification equations from each other.

The proof of equality of discrete logarithms employed by Chaum and van Antwerpen [6] resembles the Schnorr signature. It proves that $\log_g y = \log_v u$ for some u and v. Note that $[g, y]$ **or** $[u, v]$ must be *trusted* or *certified*. The verification equation for such a scheme is as follows:

$$c \stackrel{?}{=} \mathcal{H}(m, A, B)$$

where,

1. c is the message digest;
2. $\mathcal{H}$ is a secure hash;
3. m is the set of messages;
4. $[c, r]$ is the signature cipher-text; and
5. $A = y^c g^r$ and $B = u^c v^r$ are the symmetric keys.

The integrity goal of this scheme can be expressed as follows:

$$\Big(\big(([g, y] \stackrel{[c,r]}{\rightarrow} A) \wedge ([v, u] \stackrel{[c,r]}{\rightarrow} B)\big) \stackrel{c}{\rightarrow} m\Big) \tag{2}$$

The symmetric keys A **and** B provide integrity service to m. It is crucially important to note that $[g, y]$ or $[v, u]$ *must be certified* (using some private or public certification scheme) before any integrity deductions can be made. The protocol *associates* the integrity of $[g, y]$ (or $[v, u]$) with the integrity of $[v, u]$ (or $[g, y]$). Once this association is made and the *absolute* integrity of at least one of the key tuples is deduced, then the integrity of the symmetric keys $[A, B]$, and thereby the message m, can be deduced. Without certification of any of the keys, no meaningful deductions on the integrity service can be made. Note that this requirement is *inherited* from the Schnorr signature scheme represented in Equation 1.

2.2 Characterising Schnorr-Type Blind Signature Schemes

The blind signature technique [8] allows an entity to obtain a signature tuple on a message from a signer without revealing either the signature tuple or the message. This allows the entity to prove to any other entity that it was authorised by the signer without revealing its identity – the entity is anonymous.

A well known method to obtain blind signature requires the signer to engage in a honest-verifier zero-knowledge identification protocol with the receiver (of the signature), who would play the role of a *skewed honest-verifier* to obtain

the blind signature. Chaum and Pedersen [7] demonstrated the technique to obtain a blind Schnorr signature, which was later modified by Brands [4] to obtain a specialised version called restrictive blind signature. The purpose of this section is to characterise both these schemes in order to highlight their subtle and important properties, which are usually ignored by some protocol designers. This oversight introduces many deficiencies in the integrity goal of the resulting cryptosystem.

A Schnorr-type blind signature was first proposed by Chaum and Pedersen [7]. The signature tuple is the same as that of Schnorr signature scheme (see Section 2.1) and has the same signature verification equation. The only difference is that the signer *cannot know* the message that is being signed, which in the case of Schnorr signature is the symmetric key and *not the message itself*. This is a subtle point that should actually mean that the signer is authorising the symmetric key only and *does not necessarily* authorise the message that the symmetric key may provide integrity to – as was the case in the original Schnorr signature scheme. Interestingly, this problem has a counterpart in the key recovery research (and cryptologic research as a whole), where it is a difficult problem to *restrict the use of certified keys* [10].

Since the verification equation for a blind Schnorr signature is the same as the Schnorr signature scheme, this subtlety is introduced in the representation of the integrity goal by employing a *modifier*. This is because the blinding process provides *confidentiality service* and the syntax presented in this paper deals only with the *integrity service*. Since the blinding process does not alter the integrity goal of the protocol, any alteration of the representation of the integrity goal for the Schnorr signature, to introduce the subtlety, must be purely a convention. The best way to accomplish this requirement would be to introduce a modifier. In Equation 1, the message that is signed, m, is represented employing a modifier as $\overline{m}$. Syntactically, Equation 1 is otherwise unchanged. The integrity goal is represented as follows:

$$\left([g, y] \xrightarrow{[c,r]} A \overset{c}{\rightarrow} \overline{m}\right) \tag{3}$$

Note that the signature generation procedure may or may not be blinded[2], so the modifier is intended only for the interpretation of a potential weakness in argument. In other words, the modifier is *a statement of intent* and *not of a fact*. In the previous equation, the modifier suggests that the signer *may have no control over* the message, m.

The restrictive blind signature by Brands [4] is similar to the blind Schnorr signature scheme [7], with an additional property that the signer guarantees the *structure* of the symmetric key, A. In the original proposal [4], the signer employs the Schnorr variant (by Chaum and van Antwerpen, see Section 2.1) represented by Equation 2 and guaranteed the representation (structure) of one of the symmetric keys with respect to the bases $[g_1, g_2]$. The verification equations

[2] In the case of an e-cash system the customer could engage in a normal Schnorr signature protocol with the bank, and the merchant cannot discern this fact.

employed by the merchant (during the spending phase) and the bank (during the deposit phase) in Brands' scheme are as follows:

$$
\begin{aligned}
c &= \mathcal{H}(A, B, z, a, b) \\
a &\stackrel{?}{=} g^r y^{-c} \\
b &\stackrel{?}{=} A^r z^{-c} \\
d &= \mathcal{H}_0(A, B, \cdots) \\
B &\stackrel{?}{=} g_1^{r_1} g_2^{r_2} A^{-d}
\end{aligned}
$$

where:

1. c and d are message digests;
2. $\mathcal{H}$ and $\mathcal{H}_0$ are secure hash functions;
3. $[A, z]$ is a temporary key pair;
4. B is a message;
5. $[a, b]$ is the symmetric key tuple blindly *authorised* by the bank; and,
6. $[g, g_1, g_2, y, y_1, y_2]$ is the public key of the bank such that $y = g^{x_B}$, $y_1 = g_1^{x_B}$ and $y_2 = g_2^{x_B}$, where x_B is the banks private key;
7. $[r, c]$ is the signature tuple by the bank; and,
8. $[r_1, r_2]$ is the signature tuple on B employing the key$[g_1, g_2, A]$.

The integrity goal of this scheme is represented as follows:

$$
\begin{aligned}
&\left(\left(([g, y] \xrightarrow{[c,r]} a) \wedge ([A, z] \xrightarrow{[c,r]} b)\right) \xrightarrow{c} \overline{B}\right) \wedge \\
&\left([g_1, g_2, A] \xrightarrow{[r_1, r_2, d]} B \xrightarrow{d} [A, \cdots]\right) \qquad (4)
\end{aligned}
$$

It can be read as: the bank authorises the symmetric keys $[a, b]$ using its public key $[g, y]$ and, $[A, z]$ by its *association* with $[g, y]$. The symmetric keys provide integrity service to B (note the use of the modifier as $\overline{B}$ to represent the blind operation). This is the joint statement of the first verification equation. The second verification equation provides integrity service to B by employing the public key $[g_1, g_2, A]$ and the cipher-texts $[r_1, r_2, d]$. B, in turn, provides integrity service to a predetermined set of messages and A. This is not a blinded operation. The implicit assumption for the goal of this proposal is the *association* of the bases $[g_1, g_2]$ with the key A, which was a part of the key $[A, z]$ which was *associated* with $[g, y]$ by the blind signature process. Thereby, whoever possessed the signature (the first verification equation) must also possess the knowledge of the representation of A with respect to the base $[g_1, g_2]$ (just as the Schnorr signature scheme required the signer to possess the representation of the public key y with respect to the base g), and therefore the representation of B. This additional check allowed the bank (which took part in the signature generation process) to gain another implicit confidence: the blind signature transcript contains a valid, *hidden* identity that is a representation of the bases $[g_1, g_2]$. In the case of electronic cash systems employing blind signature, the merchant, without trusting the bank, *cannot* gain this knowledge as it can make no logical deductions about the withdrawal protocol (signature generation process).

3 Analysis of an Efficient E-Cash Proposal

Electronic cash systems, like physical cash systems and unlike electronic payments systems like credit cards, allow the users to anonymously spend legitimate amounts of currency. The anonymity property is mutually exclusive of the properties for tracing transactions.

This section will analyse the e-cash proposal of Radu, Govaerts and Vandewalle [12]. The proposal is a three-phased withdrawal mechanism presented briefly as follows:

1. *get_pseudonym* protocol between the user and the bank to obtain a restrictive blind signature on a pseudonym, π, by employing the Brands withdrawal protocol (see Equation 4). This allows the bank to guarantee that the pseudonym π is derived from a registered identity π_0.
2. *withdraw_big_coin* protocol between the user and the bank allows the user to obtain a blind Schnorr signature (see Equation 3) on a *big coin* that associates a pseudonym, β with a valid long-term pseudonym π; and,
3. *exchange_big_coin* protocol between the user and the bank that allows the user to *anonymously* withdraw many *small coins* after providing the bank with a valid *big coin* and the corresponding long term pseudonym π.

The user can spend the *small coins* with any merchant. Radu *et al.* proposed the use of a smart-card during the spending protocol that will act as an observer to prevent double spending of small coins (refer to the paper by Chaum and Pedersen [7] for a detailed discussion on this topic). The certified public keys of the bank is represented by the tuple, $[g, P, P_1]$ such that the bank possesses the representation of P and P_1 to the base g.

As stated previously in Section 2.2, a blind signature must be considered as an authorisation for a symmetric key and not for the message that could be serviced by the symmetric key. Radu *et al.* did not observe this caution in their proposal for an efficient e-cash. As will be shown, this oversight results in a weakness in their proposal that allows unaccounted transfer of funds between accounts, that is the property of non-transferability is not achieved.

The verification equations that the bank employs to verify the long-term pseudonym during the *exchange_big_coin* phase are as follows:

$$
\begin{aligned}
c &= \mathcal{H}(\pi, z, A, B) \\
A &\stackrel{?}{=} g^r P^c \\
B &\stackrel{?}{=} \pi^r z^c \\
d &= \mathcal{H}(\beta, \alpha) \\
\alpha &\stackrel{?}{=} g_1^{r_1} g_2^{r_2} \pi^d
\end{aligned}
$$

These are the verification equations of Brands' restrictive blind signature scheme discussed in Section 2.2. The representation for the verification of long-term pseudonym component of the big-coin is as follows:

$$\Big(\big(([g,P] \xrightarrow{[c,r]} A) \wedge ([\pi,z] \xrightarrow{[c,r]} B)\big) \xrightarrow{c} \overline{[\pi,z]}\Big) \wedge$$
$$\Big([g_1,g_2,\pi] \xrightarrow{[r_1,r_2,d]} \alpha \xrightarrow{d} [\beta]\Big) \tag{5}$$

In the Brands' scheme, the symmetric key α (B in Equation 4) was serviced by c, which restricted the use of α to only one servicing – otherwise the private key of the user would be revealed (a deficiency of Schnorr-type signature schemes). Whereas, in the scheme proposed by Radu *et al.*, the symmetric key α was not serviced by c. Thereby the value for α can be changed (mutable) to allow for multiple servicing of multiple values of β by π.

The verification equation that the bank employs to verify the big coin during the *exchange_big_coin* phase are as follows:

$$e = \mathcal{H}(\beta, \pi, D)$$
$$D \stackrel{?}{=} g^{r_3} P_1^e$$

This is a blind Schnorr signature explained in Section 2.2 by Equation 3. The representation for the verification of the short-term pseudonym (β) component of the big-coin is as follows:

$$\Big([g,P_1] \xrightarrow{[e,r_3]} D \xrightarrow{e} \overline{[\beta,\pi]}\Big) \tag{6}$$

Note that the claimed association between a long term pseudonym, π, and the short term pseudonym, β, happens during this protocol. Also, note the modified term, $\overline{[\beta,\pi]}$, which suggests that the signer (the bank) with the public key $[g,P_1]$ *can have no control over* the values $[\beta,\pi]$.

Radu, Govaerts and Vandewalle analysed $[e,r_3]$ as a *signature* on $[\beta,\pi]$ by the key pair $[g,P_1]$, the certified public key of the bank. Therefore, they argued that association was authorised by the bank. The flaw in this argument is: $[e,r_3]$ is a *blind signature* on $[\beta,\pi]$. Referring to equation 6, clearly the integrity check relies on the *use of the key*, D, which was authorised by the bank, to associate the tuple $[\beta,\pi]$ and this problem is similar to the generic situation explained in Section 2.2. That is, the bank is *trusting* the user to correctly associate one of his/her long-term pseudonyms, π, with a short-term pseudonym, β. This allows the user to associate the π value of another user with the β value that resulted from his/her withdrawal. In effect, this would allow *unaccounted* money transfer between users, which may result in perfect black-mailing and/or money-laundering [18]. Although Radu *et al.* did not comment about the property of *non-transferability*[3] in their paper, many practical monetary systems require this property for their proper functioning. Therefore, their scheme lacks the *non-transferability* property, primarily due to the lack of consistent integrity checks.

In order to visualise this problem let the long-term pseudonym of a blackmailer be π, which was derived from his/her long term identity π_0 using the

[3] The property which is essential to prevent unaccounted transfer of funds.

get_pseudonym protocol. The black-mailer can perform the following actions to achieve a perfect-blackmail;

1. allow the victim user to participate in the mutual-authentication protocol that takes place before the *withdraw_big_coin* transaction;
2. logically or physically hijack the withdrawal terminal from the victim user to prevent him/her from registering the value of π; and,
3. perform the *withdraw_big_coin* transaction with the bank as prescribed by the protocol, employing π as the pseudonym.

4 Analysis of the Binding ElGamal Proposal

Key recovery infrastructures aim to provide *restricted* confidentiality channel for users communications. The confidentiality property of the channel is restricted because, unlike the traditional key establishment systems, the messages communicated by the users can be *accessed* or *wire-tapped* by *authorised* entities called escrow agents. Such systems were primarily motivated by the needs of law enforcement agencies.

Verheul and van Tilborg [17] proposed a fraud detectable key recovery scheme. The proposal was aimed to allow any third party to verify if a sender has encrypted the session key value to the receiver and the escrow agents. The verification equations, which were proposed to detect activities that could by-pass the key-recovery infrastructure, were:

$$\begin{aligned} c &= \mathcal{H}(E, C, R_A, R_B, R_M, D, F, I, \cdots) \\ D &\stackrel{?}{=} g^c C^r \\ F &\stackrel{?}{=} (y_A/y_M)^c (R_A/R_M)^r \\ I &\stackrel{?}{=} (y_B/y_M)^c (R_A/R_M)^r \end{aligned} \tag{7}$$

where: $\mathcal{H}$ is a secure hash function, $[c, r]$ is a Schnorr signature tuple. This check was aimed to show that the message encrypted in $R_A = Sy_A^k$ and $R_M = Sy_M^k$ ($C = g^k$) is the same, *without revealing the message*.

Using the notation presented in Equation 2, Section 2.1, the following representation for the verification equations of the key recovery scheme can be determined:

$$\begin{aligned} &\Big(((([g, C] \stackrel{[c,r]}{\rightarrow} D) \wedge \\ &([y_A/y_M, R_A/R_M] \stackrel{[c,r]}{\rightarrow} F) \wedge \\ &([y_B/y_M, R_B/R_M] \stackrel{[c,r]}{\rightarrow} I)) \stackrel{c}{\rightarrow} [E, C, R_A, R_B, R_M, \cdots]\Big) \end{aligned} \tag{8}$$

Note that none of the key pairs ([g, C], [$y_A/y_M, R_A, R_M$] [$y_B/y_M, R_B/R_M$]) providing integrity service are certified. It is evident that this representation is similar to the representation provided in Equations 1 and 2. By comparing the above representation with Equations 1 and 2, the following observations can be made:

1. none of the key pairs ($[g, C]$, $[y_A/y_M, R_A/R_M]$ and $[y_B/y_M, R_B/R_M]$) can be trusted because they are uniformly chosen by the sender (who is not trusted for certification procedures);
2. ratios of keys provide the integrity service to the symmetric keys F and I, which is not a *standard* assumption of Schnorr-type signatures.

These observations suggest a deficiency in the system that allows the sender to manipulate the keys, which were meant to be the *starting point* of the integrity service – that is if the starting point is corrupted then the integrity service that it transfers is also corrupted. This weakness in the integrity service could potentially result in attacks on the protocol, like the attack to be presented in this section.

Prior to discussing an attack on the key recovery system, the meaning of a non-trivial attack must be understood. A key recovery protocol is deficient if successful adversaries abide with the message formats suggested by the protocol and *procure legitimate services from the key recovery infrastructure* to ensure secure communication. For example, if a public-key based key recovery system provides robust certification mechanism, such as robust public key infrastructures, and requires key recovery enablement before the certification can be employed, then an adversary is successful when certified public keys are employed and key recovery is avoided. The attack on the proposal, by Verheul and van Tilborg [17], by Pfitzmann and Waidner [11] need *not necessarily* be an attack on the protocol proposed by Verheul and van Tilborg, rather it is an attack on all session-key recovery systems without any form of private-key recovery. It outlines the *generic* concealed-encryption attack[4] on key recovery protocols and *fails* to explain the manner in which the *concealed key* may be established. Although the attack proposed in this section exploits the property of concealed-encryption attack, it is not a generic attack on all session-key recovery protocols, rather it is a specialised attack on the proposal [17], *which resulted from an oversight in the protocol design*. Moreover, this section will detail the manner in which an illegal session key can be established using *the key recovery infrastructure*. This distinction is important for protocol designers, who may employ the proposed fraud detection mechanism [17] for a different application that may not have properties similar to that of key-recovery applications. For example, refer to the paper by Abe [2], which successfully employed a similar integrity verification mechanism for a mix network proposal.

Suppose that the sender and a hidden receiver ($\tilde{M}$) would like to communicate using the actual receiver (M) as the decoy. The sender can accomplish this by employing the following steps:

1. Choose a random session key, $\tilde{S}$.
2. Encrypt the message with $\tilde{S}$ to obtain the cipher-text, E.
3. Obtain the public keys of the hidden receiver, y_H, the decoy, y_M and the authorities (y_A, y_B).

[4] There is no technique available to check if a claimed key was used during the encryption process — verifiable encryption for symmetric key systems is not currently available

4. Choose a random value for k.
5. Compute a decoy session key, $S = \tilde{S}y_H^k / y_M^k$.
6. Encrypt the decoy session key for the decoy and the authorities, $R_M = Sy_M^k = \tilde{S}y_H^k$, $R_A = Sy_A^k$, $R_B = Sy_B^k$ and $C = g^k$.
7. Form the verification equation as suggested by the representation in Equation 8.
8. Send the cipher-texts and verification parameters to decoy.

The hidden receiver performs the following steps:

1. Wiretap the communication to decoy to obtain E, R_M and C.
2. Obtain session key, $\tilde{S} = R_M / C^{x_H}$, where x_H is the private key of the hidden receiver.
3. Decrypt E using $\tilde{S}$ to obtain the message.

The monitor will verify the equations properly, the decoy receiver and the authorities will retrieve the decoy session key, S, from the respective cipher-texts employing the respective private keys and, the decoy session key, S, will not decrypt E correctly. Also note that it will be difficult to find the hidden receiver, y_H, or the actual session key, $\tilde{S}$ (finding the hidden receiver would imply breaking of the multi-ElGamal cryptosystem proposed in the paper [17]).

5 Conclusion

The paper presented a novel technique to represent the integrity goal of a system by accounting for all the verification equations and ignoring the unnecessary protocol complexities that produced the equations. An abstraction to encompass the *unpredictability* of the protocol participants was also proposed. The use of the technique was demonstrated by the identification of *similar* protocol deficiencies in *seemingly* different scenarios.

Many proposals for *compliant* systems tend to ignore the importance of the integrity service, while in pursuit of the confidentiality service. Blaze [3] formulated an attack on the integrity service in the Clipper proposal [16], which was predominantly focused on the confidentiality service. Unfortunately, many protocols in various fields of cryptologic application still succumb to attacks similar to those detailed in Sections 3 and 4, namely attacks exploiting weaknesses in integrity services. In order to design robust and secure protocols the integrity and the confidentiality services must be carefully designed and integrated.

Prospective formal syntax that can represent precisely both the confidentiality and the integrity goals will greatly improve protocol logic development. Research for such a syntax will be very useful, both theoretically and practically.

References

1. Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000), Sendai, Japan*, 2000. To appear.

2. Masayuki Abe. Mix-networks on permutations networks. In K. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology – ASIACRYPT'99*, volume 1716 of *LNCS*, pages 258–273. Springer-Verlag, 1999.
3. Matt Blaze. Protocol failure in the escrowed encryption standard. In *The 2nd ACM Conference on Computer and Communications Security*, November 1994.
4. Stefan Brands. Untraceable Off-line Cash in Wallet with Observers. In Tor Helleseth, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *LNCS*, pages 344–359. Springer-Verlag, 1993.
5. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. In *Proceedings of the Royal Society of London*, volume 426, pages 233–271, 1989.
6. D. Chaum and H. van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *LNCS*, pages 212–216. Springer-Verlag, 1989.
7. David Chaum and T. Pedersen. Wallet Databases with Observers. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer-Verlag, 1992.
8. David Chaum. Blind Signatures for Untraceable Payments. In Sherman A.T. Chaum D., Rivest R.L., editor, *Advances in Cryptology – CRYPTO'82*, pages 199–203. Plenum Press, 1983.
9. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, 1986.
10. Lars R. Knudsen and Torben P. Pedersen. On the difficulty of software key escrow. In U. M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *LNCS*, pages 237–244. Springer-Verlag, 1996.
11. Birgit Pfitzmann and Michael Waidner. How to break fraud-detectable key recovery. *Operating Systems Review, ACM press*, 32(1):23–28, January 1998.
12. Cristian Radu, René Govaerts, and Joos Vandewalle. Efficient electronic cash with restricted privacy. In Rafael Hirschfeld, editor, *Financial Cryptography, FC'97*, volume 1318 of *LNCS*, pages 24–28. Springer-Verlag, 1997.
13. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
14. Rainer A. Rueppel. A formal approach to security architectures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *LNCS*, pages 387–398. Springer-Verlag, 1991.
15. C.P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4:161–174, 1991.
16. U.S. DEPARTMENT OF COMMERCE / National Institute of Standards and Technology. *Federal Information Processing Standard 185—Escrowed Encryption Standard*, February 1994.
17. Eric R. Verheul and Henk C.A. van Tilborg. Binding ElGamal: A fraud-detectable alternative to key-escrow proposals. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *LNCS*, pages 119–133. Springer-Verlag, 1997.
18. B. von Solms and D. Naccache. On Blind Signatures and perfect crimes. *Computers and Security*, pages 581–583, October 1992.