The European Online Magazine for IT Professional
http://www.upgrade-cepis.org
Edizione italiana a cura di ALSI e Tecnoteca
http://upgrade.tecnoteca.it

Firma Digitale e Crittografia con XML

di
Antonio F. Gòmez-Skarmeta, Marìa-Encarnaciòn Martìnez-Gonzàlez, Eduardo Martìnez-Gracià e Gregorio Martìnez-Pèrez
(Traduzione italiana, a cura di Luigi Caso (ALSI), dell'articolo "Digital Signature and Encryption with XML", pubblicato sul Vol. III, No. 4, Agosto 2002, della rivista online UPGrade, a cura del CEPIS)

Parole chiave: XML, Firma Digitale, Cifratura, W3C, IETF, ebXML, SOAP, x-SPEED.

1 Introduzione

Oggigiorno, XML sta diventando un efficace meccanismo per lo scambio di dati attraverso Internet, e l'e-commerce è una delle aree in cui XML sta giocando un ruolo decisivo. Un aspetto importante da tenere in considerazione in uno scenario di e-commerce è la fornitura della sicurezza nelle comunicazioni, in modo da evitare che persone non autorizzate accedano o modifichino informazioni sensibili o confidenziali. Come è possibile aggiungere la sicurezza ad XML? Una maniera semplice per farlo è quella di considerare un documento XML come un normale documento binario ed applicargli un qualsiasi meccanismo di crittografia per firmare o cifrare documenti. Ma questo metodo non è flessibile, perchè consente solo di firmare o cifrare interi documenti XML. Cosa accade, ad esempio, se si vuole cifrare parti differenti di un documento per inviarli a differenti destinatari, o firmare porzioni di un documento da parte di mittenti diversi, o applicare firma e crittografia contemporaneamente a parti di un documento in un ordine qualsiasi?
È possibile trovare risposta a tali quesiti negli standard emergenti *XML Signature* ed *XML Encryption*, entrambi sviluppati congiuntamente dall'IETF e dal W3C. Questi standard descrivono come applicare firme digitali e crittografia all'XML in un modo davvero flessibile e potente.

2 XML Digital Signature

XML Digital Signature [XML-DSIG] può essere applicato a qualsiasi struttura dati, inclusi documenti binari, interi file XML e frammenti di dati. Inoltre, può essere applicato contemporaneamente al contenuto di diversi documenti.

Nel firmare dati XML, c'è un problema importante da tenere in mente: la flessibilità dell'XML. Per definizione, XML consente di esprimere le stesse informazioni in modi diversi, così che due documenti XML possono essere logicamente equivalenti, ma possono avere differenze meno importanti che non riguardano la loro struttura logica, come gli spazi all'interno dei tag, delimitatori di linea, commenti, e così via. Le applicazioni XML tendono ad ignorare quelle differenze perchè non hanno impatto sull'informazione in sè, ma tutto ciò è cruciale per le firme digitali perchè la validazione di una firma digitale dovrebbe essere applicata esattamente sullo stesso flusso di byte di quando è stata generata.

Per risolvere questo problema, è stato introdotto il concetto di *canonizzazione* (*canonicalization*) dei documenti XML [XML-C14N]: convertire un documento XML nella sua forma canonica consiste nell'applicargli un insieme di trasformazioni in modo da rappresentare le informazioni variabili in maniera standard. Così, due documenti logicamente equivalenti avranno esattamente la stessa forma canonica.

Oltre a questo, è importante ricordare che ci sono differenti tipi di XML signature. Si parla di *enveloping signature* quando la firma contiene l'oggetto firmato, di *enveloped signature* se l'oggetto firmato contiene la firma e di *detached signature* quando l'oggetto firmato non è incluso nell'elemento firma.

Per creare una signature XML per un qualsiasi contenuto digitale si deve, per prima cosa, ottenere un valore "digest" per ciascun oggetto (dato) che si vuole firmare. Poi, l'insieme risultante di questi valori viene inserito in un elemento XML, insieme ad altre informazioni che verranno dettagliate in seguito in questo articolo, ed infine viene generato un valore "digest" dell'elemento appena descritto e viene firmato con una chiave privata. In questo modo, il processo di validazione prevede due passi: prima, viene validata la firma stessa, assicurando l'integrità del valore "digest" dell'intero elemento firmato, poi, viene validato il valore "digest" di ciascun dato.

Le firme digitali XML sono rappresentate dall'elemento *Signature*, che ha la struttura visibile nella Figura 1. Si userà "?" per indicare zero o una occorrenza di un elemento, "+" per indicare una o più occorrenze e "*" per indicare zero o più occorrenze.

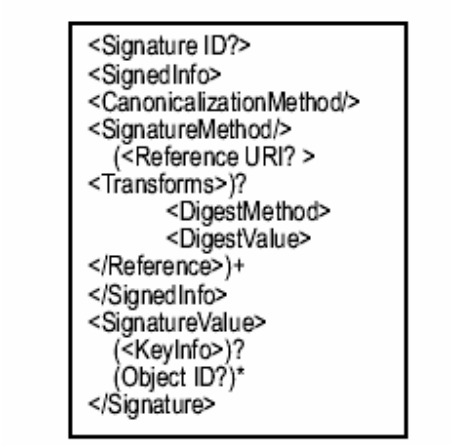

Figura 1: La struttura generica dell'elemento *Signature*

Come si vede, l'elemento *Signature* contiene quattro sotto-elementi: *SignedInfo*, *SignatureValue*, *KeyInfo* ed *Object*. L'elemento *SignedInfo* rappresenta le informazioni realmente firmate. Questo elemento contiene anche informazioni sul metodo di *canonizzazione* e sull'algoritmo di firma usati per calcolare il campo *SignatureValue*. *SignedInfo* contiene un elemento *Reference* per ciascun oggetto firmato, che include un riferimento all'oggetto (attraverso un URI, Uniform Resource Identifier), il metodo usato per calcolare il valore "digest" dello stesso ed il corrispondente valore "digest". Un elemento *Reference* può anche contenere delle trasformazioni che, se applicate all'oggetto identificato, producono l'input per l'operazione di calcolo del valore "digest" (un esempio di tali trasformazioni è il metodo di *canonizzazione* spiegato in precedenza).

L'elemento *KeyInfo* indica la chiave da usare per generare (e validare) la firma. Può includere le chiavi, i nomi delle chiavi, i certificati ed ogni altra informazione utile per il processo di crittografia. Infine, l'elemento *Object* viene usato per includere qualsiasi altro oggetto all'interno dell'elemento *Signature* e può essere firmato o meno. La Figura 2 mostra un esempio di una firma digitale XML.

3 XML Digital Encryption

Simile ad XML Signature, *XML Encryption* [XML-Enc] può essere applicato a qualsiasi struttura dati, interi documenti XML o semplici "content element" XML. Dopo un processo di XML encryption, si ottiene un elemento *EncryptedData* che contiene, tramite un sotto-elemento o un riferimento, i dati da cifrare. Se questi dati costituiscono un elemento XML, *EncryptedData* sostituirà quest'ultimo nella versione cifrata del documento XML finale. La Figura 3 fornisce una vista più approfondita della struttura dell'elemento *EncryptedData*.


```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference URI="http://www.um.es/mydocument/document.xml">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>j6lwx3rvEPO0vKtMup4N</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
      </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

Figura 2: Un esempio di firma digitale XML

Come è possibile vedere in tale figura, *EncryptedData* contiene quattro elementi principali. Il primo di essi, *EncryptionMethod*, indica l'algoritmo di cifratura usato, mentre il secondo, *ds:KeyInfo*, identifica la chiave usata per cifrare i dati. *CipherData* contiene i dati cifrati o un riferimento alla loro posizione ed, infine, l'elemento opzionale *EncryptionProperties* fornisce altre informazioni sulla generazione dei dati cifrati, ad esempio un timestamp.

```
<EncryptedData Id? Type?>
  <EncryptionMethod/>?
  <ds:KeyInfo>
    <EncryptedKey>?
    <AgreementMethod>?
    <ds:KeyName>?
    <ds:RetrievaMethod>?
    <ds:*>?
  </ds:KeyInfo>?
  <CipherData>
    <CipherValue>?
    <CipherReference URI?>?
  </CipherData>
  <EncryptionProperties>?
</EncryptedData>
```

Figura 3: La struttura generica dell'elemento *EncryptedData*

È anche importante dire che, in alcuni casi, gli elementi *EncryptedMethod* e *ds:KeyInfo* possono essere opzionali, in quanto il mittente ed il destinatario potrebbero essersi accordati in anticipo sull'algoritmo di cifratura e sulla chiave. L'elemento *ds:KeyInfo* offre modi differenti di fornire la chiave di cifratura, sia cifrata in un elemento *EncryptedKey*, sia attraverso un sistema di gestione delle chiavi – per es. Diffie-Hellman – tramite un elemento *AgreementMethod*; altri modi ancora sono fornire l'identificativo di una chiave nota nell'elemento *ds:KeyName* o il

riferimento alla posizione dove la chiave è memorizzata – come nei sistemi DNSsec – tramite l'elemento *ds:RetrievalMethod*.

Nella Figura 4 si può vedere un esempio di un elemento cifrato usando una chiave simmetrica. In questo documento XML l'elemento di testo in chiaro viene sostituito con l'elemento *EncryptedData*, come spiegato in precedenza.

```
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'type='http://www.w3.org/2001/04/xmlenc#Element'/>
  <EncryptionMethod Algorithm='http://www.w3.org/2001/04/xmlenc#3des-cbc'/>
  <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
    <ds:KeyName>My Key</ds:KeyName>
  </ds:KeyInfo>
  <CipherData>
    <CipherValue>DEADBEEF</CipherValue>
  </CipherData>
</EncryptedData>
```

Figura 4: Un esempio di cifratura XML

In questo esempio è stato usato l'attributo *Type* per indicare che il dato cifrato è un elemento. L'algoritmo applicato è *3DES-CBC* ed usa una chiave simmetrica, chiamata *My Key*, per cifrare i dati. L'elemento *CipherData* contiene la sequenza dei byte cifrati, usando la codifica *base64*, all'interno del suo sotto-elemento *CipherValue*.

4 La combinazione degli standard XML Signature ed XML Encryption

L'applicazione di entrambi gli standard, *XML Signature* ed *XML Encryption*, su parti di un documento XML può determinare delle ambiguità riguardo a come effettuare la successiva decifratura e/o il processo di verifica.

Infatti, quando si verifica una firma, bisognerebbe sapere se questa è stata calcolata sulla rappresentazione dei dati cifrata o su quella non cifrata. Oltre a questo, se sono state effettuate delle operazioni di cifratura su contenuti già firmati, sarà necessario, prima della verifica, decifrare quelle parti che erano state cifrate dopo la firma.

Per indicare la corretta sequenza di decifratura/verifica quando si usano contemporaneamente XML signature ed encryption, è stata proposta una nuova XML signature transform [XML-DSIG-Decrypt], denominata *decryption transform*. Questo elemento prende come parametro in input una lista di riferimenti a delle porzioni di dati che sono state cifrate prima di essere firmate. Questo metodo implica che, quando il destinatario trova una *decryption transform* all'interno di un elemento di firma, decifrerà tutti i contenuti cifrati del documento, cioè quelli che sono referenziati nella lista inclusa nella *decryption transform*.

5 Lo standard ebXML

Una delle iniziative più interessanti tra gli standard per l'e-commerce che usano XML signature è *ebXML* [ebXML], proposto da OASIS e UN/CEFAT. *ebXML* usa SOAP [SOAP], Simple Object Access Protocol, come sottosistema di messaging. SOAP è uno standard W3C, che definisce le specifiche di packaging dei messaggi usando l'XML; i messaggi vengono trasportati da protocolli come HTTP o SMTP, utilizzando i tipi MIME per identificarli correttamente. La Figura 5 mostra la struttura XML di un messaggio SOAP.

In *ebXML* l'interazione tra "business information systems" è basata sull'utilizzo di un registro centrale che agisce da directory. Le aziende presentano a questo registro un documento, chiamato Collaboration Protocol Profile (CPP), che individua i propri "business processes and interfaces" (simile al WSDL, Web Services Description Language, per i web service). In questo modo il tipo di "business" tra aziende viene determinato dinamicamente confrontando i rispettivi CPP. Questo processo genera un documento chiamato Collaboration Protocol Agreement (CPA), che viene opportunamente registrato. Il processo di trasmissione dei messaggi viene effettuato usando il servizio di messaggistica di *ebXML*, definito sulla base di pacchetti SOAP. Per rendere chiari questi concetti, la

Figura 6 mostra un pacchetto SOAP che trasporta informazioni *ebXML* con una firma. Le XML signature compaiono all'interno dell'header del messaggio SOAP, mentre gli oggetti firmati sono localizzati in un'altra parte del messaggio "multipart", referenziati dall'elemento *eb:Manifest* del corpo del messaggio SOAP.

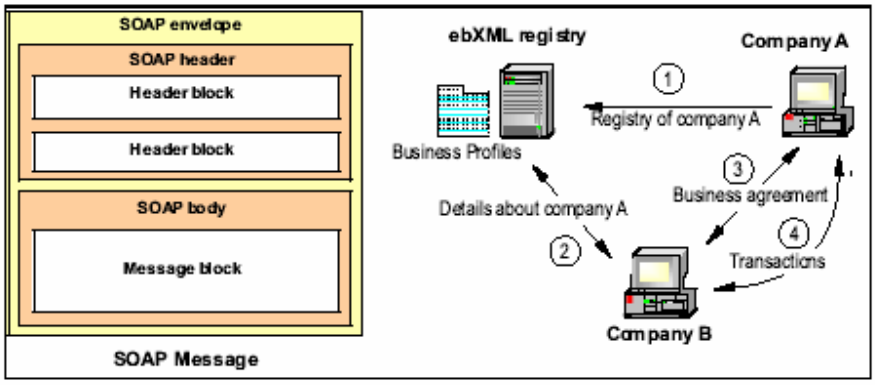

Figura 5: Il formato di un messaggio SOAP, e l'interazione di base con ebXML

6 Un nuovo protocollo di pagamento: x-SPEED

In questo paragrafo verrà descritto un protocollo di pagamento che è il risultato di un lavoro di ricerca realizzato dal nostro gruppo di lavoro (Universidad de Murcia, ndr). Questo protocollo, chiamato SPEED, ossia *Smartcard-based Payment with Encripted Delivery* [Ruiz-Martìnez et al. 2001], fornisce un sistema di pagamento basato su borsellino elettronico implementato con smart-card e distribuzione cifrata di prodotti elettronici.

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0"?>
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <eb:MessageHeader>
    </eb:MessageHeader>
    <Signature>
      <SignedInfo>
        <Reference URI="cid:xxx">...</Reference>
          </SignedInfo>
    </Signature>
  </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <eb:Manifest eb:id="Mani01" eb:version="2.0">
      <eb:Reference xlink:href="cid:xxx
                    xlink:role="http://ebxml.org/gci/invoice">
      </eb:Reference>
    </eb:Manifest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-ID: xxx

<Invoice>
  <InvoiceData>...</InvoiceData>
</Invoice>
```

Figura 6: Un messaggio SOAP con contenuto ebXML con firma digitale

In poche parole, una transazione SPEED trasferisce beni elettronici da un rivenditore ad un cliente, addebitandoli sul suo borsellino elettronico ed incrementando il saldo del conto del rivenditore del valore del prodotto acquistato. SPEED definisce un insieme di fasi, includendo negoziazione del prezzo, consegna del prodotto e pagamento. Inoltre, ci sono due possibili modi di operare: il modo normale (*normal mode*), che consente un processo di negoziazione del prezzo del prodotto ed è stato progettato per fornire un maggior numero di aspetti di sicurezza (ad esempio, prevenzione di attacchi di tipo DoS e completa autenticazione delle entità che prendono parte alla transazione) e, all'opposto, il modo veloce (*fast mode*), che ha meno messaggi del modo normale ed è concepito per essere usato per l'acquisto di beni più piccoli o per scenari con minori esigenze di sicurezza.

Il progetto e l'implemetazione iniziali erano basati sull'utilizzo di alcuni degli standard più importanti riguardanti le comunicazioni sicure. Alcuni di essi, come X509v3 (che veniva usato per identificare le parti in uno scenario di acquisto) e WG10 [WG10 1996] (standard per il borsellino elettronico), rimangono nella nuova versione XML del protocollo. Altri, invece, sono stati sostituiti in modo da ottenere una specifica XML più completa e più estensibile dei messaggi del protocollo, con lo stesso livello di sicurezza. Questo è il caso di ASN.1, usato per specificare la struttura dei messaggi, e PKCS#7, utilizzato come formato di crittografia di base per lo scambio di informazioni protette.

I messaggi del protocollo x-SPEED, che è l'adattamento di SPEED all'XML, sono un insieme di messaggi espressi come documenti XML con un formato ed un contenuto ben definiti. Tali messaggi vengono scambiati tra le parti coinvolte in ogni transazione. È un ambiente B2C (Business-to-Consumer), in cui sono coinvolti tre partecipanti: un cliente, un rivenditore ed un broker (una terza parte, di fiducia, che effettua i risarcimenti economici e risolve le controversie all'interno di questo modello di business). Queste parti si scambiano "business information" applicando le XML signature ai documenti. Il protocollo fornisce servizi di integrità, autenticazione e "non-repudiation" (la certezza che il documento non sia stato manomesso, ndr) ed, inoltre, la flessibilità dell'XML signature consente di firmare esattamente la parte di documento cui si è interessati, in modo chiaro e conciso. D'altra parte, l'applicazione della XML encryption alle transazioni x-SPEED fornisce la riservatezza e, come nel caso della firma, è possibile cifrare esattamente le parti che interessano.

La Figura 7 mostra la struttura generica di base di un messaggio x-SPEED, prima di applicare la cifratura.

```
<x-SPEED>
  <messageName>
    <messageNameContent>
      ...
    </messageNameContent>
    <Signature>
      ...
    </Signature>
  </messageName>
</x-SPEED>
```

Figura 7: Un messaggio x-SPEED prima di applicare la cifratura

Sostituiamo la stringa *messageName* con il nome reale del messaggio effettivo. Ciascun messaggio consiste fondamentalmente del suo contenuto, rappresentato dall'elemento *messageNameContent* (che conterrà i dati di protocollo da scambiare tra le parti), e di una firma applicata su tale contenuto, rappresentata dall'elemento *Signature*. Questa firma sarà calcolata utilizzando la chiave privata del mittente o utilizzando una chiave simmetrica concordata tra mittente e destinatario.

Per fornire riservatezza, nel complesso, al documento mostrato in Figura 7, x-SPEED creerà un nuovo documento, mostrato in Figura 8.

Figura 8: Un messaggio x-SPEED dopo l'applicazione della cifratura

In questo nuovo documento, l'elemento radice sarà l'elemento *EncryptedData* della specifica di *XML Encryption*, già spiegata nel paragrafo di quest'articolo ad essa dedicato.

La struttura di base di una transazione x-SPEED, appena vista, è una semplificazione del vero tipo di messaggio di questo protocollo. Infatti, nella maggior parte dei casi è necessario cifrare in modo selettivo parti differenti per destinatari differenti, o avere nello stesso documento parti firmate da mittenti diversi, combinando firma e cifratura contemporaneamente sugli stessi dati. Le specifiche di *XML Signature* ed *XML Encryption* permettono l'espressione di tutte queste possibilità e combinazioni in un modo semplice, chiaro, estensibile e non ambiguo, così che qualsiasi applicazione che riceve un messaggio x-SPEED saprà esattamente quali parti sono firmate, quali cifrate e qual è l'ordine di decifratura/verifica.

Riferimenti

[Cánovas-Reverte et al. 2001]
Óscar Cánovas Reverte, Antonio F. Gómez Skarmeta, and Gregorio Martínez Pérez, "PISCIS: Comercio Electrónico Basado en Infraestructuras de Certificación Avanzadas y Sistemas de Tarjeta Inteligente", Ist Spanish Conference on E-Commerce (SNE'01), Málaga, Spain, October 2001

[ebXML]
ebXML web site <http://www.ebxml.org>

[Housley et al. 1999]
R. Housley, W. Ford, and D. Solo, "Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile", Request for Comments (RFC) 2459, IETF, January 1999

[Ruiz Martínez et al. 2001]
Antonio Ruíz Martínez, Óscar Cánovas Reverte, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. "SPEED protocol: Smartcard-Based Payment with Encrypted Electronic Delivery", Information Security Conference 2001 – ISC'01, Málaga, Spain, October 2001

[SOAP]
Simple Object Access Protocol web site <http://www.w3.org/TR/SOAP>

[WG10 1996]
CEN/TC224/WG10, "Inter-sector Electronic Purse, Part 2: Security Architecture", prEN 1546-2, January 1996

[XML-C14N]
Canonical XML <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>

[XML-DSIG]
XML Signature Syntax and Processing <http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/>

[XML-DSIG-Decrypt]
Decryption Transform for XML Signature <http://www.w3.org/TR/2002/CR-xmlenc-decrypt-20020304>

[XML-Enc]
XML Encryption Syntax and Processing <http://www.w3.org/TR/2002/CR-xmlenc-core-20020304/>

Note biografiche sugli autori

Antonio F. Gòmez-Skarmeta, laureato in Computer Science all'Universidad de Granada (Spagna), ha ricevuto in seguito il dottorato per la stessa materia dall'Universidad de Murcia (Spagna). Dal 1993 è professore di ruolo nel dipartimento di Information e Comunications Engineering dell'Universidad de Murcia. Ha lavorato in diversi progetti di ricerca sia nell'area della intelligenza artificiale distribuita (progetto M2D2) che in quella dell'e-commerce (PISCIS), così come nel tele-teaching e collaborative work (PUPITRE-NET) e nei nuovi servizi avanzati in banda larga (SABA). Attualmente partecipa a diversi progetti europei nel campo del tele-teaching e distance learning, COLAB e ITCOLE, come anche al progetto di implementazione per l'IPv6, EURO6IX, nel quale coordina gli aspetti di sicurezza. Ha pubblicato più di 40 articoli a livello internazionale. <skarmeta@dif.um.es>
Marìa-Encarnaciòn Martìnez-Gonzàlez è Computer Science Engineer, titolo acquisito presso l'Universidad de Murcia (Spagna). Attualmente sta svolgendo attività di ricerca e sviluppo presso la stessa Università. Le aree di suo maggiore interesse sono XML e sicurezza nelle comunicazioni. <encarna@dif.um.es>
Eduardo Martìnez-Gracià è Computer Science Engineer, titolo acquisito presso l'Universidad de Murcia (Spagna). Attualmente è assistente presso il dipartimento di Information e Comunications Engineering della stessa Università. Il suo lavoro di ricerca è incentrato sul campo delle applicazioni distribuite applicate al tele-learning, includendo XML, CORBA e QoS in applicazioni multimediali di trasmissione dati. <edumart@dif.um.es>
Gregorio Martìnez-Pèrez, laureato come Computer Science Engineer all'Universidad de Murcia (Spagna). Nel 1997 ha iniziato a lavorare per i servizi IT della stessa Università su diversi progetti relativi alla sicurezza nelle comunicazioni. Nel 1999 ha cominciato a lavorare come ricercatore nel dipartimento di Information e Comunications Engineering dell'Universidad de Murcia, divenendo poi insegnante nello stesso dipartimento nel 2001. La sua attività di ricerca è centrata su infrastrutture di sicurezza, carte intelligenti, sistemi di controllo degli accessi e la nuova versione del protocollo IPv6, tutti argomenti su cui ha lavorato e continua a lavorare nell'ambito di diversi progetti nazionali ed internazionali. Ha anche avuto pubblicazioni, a livello nazionale e internazionale, in conferenze e giornali. <gregorio@dif.um.es>

Luigi Caso, Laurea nel 1989 all'Università di Salerno in Scienze dell'Informazione. Attualmente lavora presso la Delos S.p.A., società del gruppo Getronics, come SW Engineer. Ha ottenuto Certificazioni Microsoft (MCP e MCSD), e partecipa, come Team Leader, alle fasi di analisi, progettazione ed implementazione di progetti SW in diversi ambiti (Gestione Documentale, Banking, Help Desk/CRM) (luigi.caso@getronics.com).