# Security Protocol Composition for Web Services

Genge Béla and Haller Piroska
"Petru Maior" University of Târgu Mureş
Department of Electrical Engineering
Nicolaie Iorga St., No. 1, 540088, Romania
{bgenge,phaller}@engineering.upm.ro

## Abstract

*The composition method, security protocol specification and middleware, developed in our previous work, are used in the process of composing Web services that use heterogeneous security protocols. Existing solutions rely on using standard security protocols that must be implemented by all Web services for the composition to be successful. This way, services implementing new protocols can not be composed with existing ones. This is why, the main contribution of the paper is the integration of automated security protocol composition methods in the field of Web services. Based on this middleware, we present a new video surveillance system, where services are automatically composed and security protocols are automatically implemented.*

## 1. Introduction

Security protocols are "communication protocols dedicated to achieving security goals" (C.J.F. Cremers and S. Mauw) [1] such as confidentiality, integrity or availability. Achieving such security goals is made through the use of cryptography. The explosive development of today's Internet and the technological advances made it possible to implement and use security protocols in a wide range of applications such as sensor networks, electronic commerce, routing environments and Web services.

In this paper we propose to use the composition methods developed in our previous work [2] in the field of Web service composition. The composition process ensures that new services are created from several initial services. There is a great amount of literature dealing with the composition of service capabilities [3, 4]. However, aspects related to the composition of security properties are not handled, services must implement the same security protocols. Over the years, there have been several security protocols proposed for ensuring inter-domain propagation of security tokens [6, 7]. Using this approach, services implementing heterogenous security protocols can not be composed.

In order to facilitate the composition of Web services that implement heterogenous security protocols, we propose a new middleware. The proposed middleware makes sure that both Web service capabilities and security protocols are composed. For each Web service we provide a description of its capabilities, with preconditions and effects and a description of the sequence of security protocols it implements. Security protocols are described using specifications constructed from existing technologies: WSDL-S [8] and OWL [9].

In order to prove that the proposed middleware can be used in the composition process of heterogenous Web services, we developed a new video surveillance system. The proposed system consists of 3 types of services: capturing services, saving services and replay services. The role of capturing services is to grab frames from physical cameras. These can be sent directly to users to be viewed live, or they can be saved by a saving service. In order for these services to communicate, their capabilities and security protocols must be composed. The role of the replay service is to replay saved data to users.

## 2 Composition

The role of the composition process is to create new, composed services, with an accumulated set of properties. These properties include service capabilities, and, as a novelty introduced by the paper, composition of security protocols. In this section we present our proposal for solving the composition problem of Web services implementing heterogeneous security protocols.

### 2.1 Composition of service capabilities

This type of composition must ensure that the capabilities provided by services can be combined with the capabilities of other services.

Because the main goal of this paper is the composition of security protocols, the composition of service capabilities is handled simply by using the preconditions and effects defined for each service. If the reader is interested in other, more complete composition methods that also involve destructive properties or data flow, these are available in the literature [3, 4].

For each service, we define a set of preconditions and effects. Let $\varepsilon$ be the set of preconditions of a service that denotes the type of data that can be received. Also, let $\varepsilon'$ be the set of effects corresponding to a composed service, denoting the type of data generated by the service. Then, for a given service $S_R$, preconditions and effects are represented as $\varepsilon S_R \varepsilon'$.

For $N$ services, the composition of service capabilities reduces to finding the sequence for which $\varepsilon'_i = \varepsilon_{i+1}$, where $i = \overline{1, N-1}, N \geq 2$ and $\varepsilon_1 = \phi$. This means that the data corresponding to service preconditions must be available for each service and the set of preconditions corresponding to the first service must be empty, denoted by $\phi$ in the previous equations.

### 2.2 Composition of security protocols

The composition of security protocols must ensure an accumulation of security properties and must have a non-destructive effect on the security properties of the original protocols. This is not a trivial task, even when there is a human factor available [11, 12]. However, for the composition of security protocols in an on-line environment, the human factor can not be present, meaning that the entire composition process must be automatic.

Such a composition method was proposed by the authors in their previous work [2]. The key to eliminate the human factor is using an enriched protocol model, with the drawback that modifying security protocols in case of unsatisfied conditions is not possible, at least for now. This means that if services implement protocols with destructive security properties, the composition will not be successful.

The composition involves two phases: composition of preconditions and effects (i.e. *PE-composition*) and the composition of protocol chains (i.e. *PC-composition*). The first phase establishes if the knowledge required by protocol participants to run a given protocol, expressed through the form of precondition predicates, is available and if the set of precondition and effect predicates is non-destructive. The second phase establishes if the composed protocol chains are non-destructive on the initial security properties.

The PE-composition uses two predicates: *PART_PREC* : $\mathsf{T}^* \times \mathsf{PR\_CC}^* \times \mathsf{PR\_CC}^*$, to verify if knowledge is available to participants, and *PART_NONDESTR* : $\mathsf{PR\_CC}^* \times \mathsf{PR\_CC}^* \times \mathsf{PR\_CC}^*$, to verify if precondition-effect pairs are non-destructive, where $\mathsf{PR\_CC}$ denotes the set of preconditions and effects. These are applied on all participant model pairs.

The PC-composition uses a canonical model that focuses on terms that can be verified by protocol participants. For each term the canonical model provides a corresponding syntactical representation through the use of *basic types*. These denote the terms that can be verified by protocol participants also including a representation for terms that can not be verified because of limited participant knowledge. The verification process uses these types to decide if attacks can be constructed on each protocol model by using terms extracted from the other considered protocol models.

In order to compose two participant chains these must be *instance independent* and *canonical independent*. The first condition refers to the non-destructive properties of preconditions and effects while the second condition refers to verifying the independence of the involved participant chains based on the canonical model. If protocols are independent, then they maintain their security properties when they are run in the same context. By using this property in the composition process, protocols maintain their security properties, resulting new protocols with accumulated properties.

## 3 Security protocol specifications

Constructing a new specification is not a trivial task, if we just look at the information required to be included: preconditions, effects, types, message sequences, security properties, protocol roles, etc. This task becomes even more complex when we realize that by automatically composing security protocols we get new protocols that must be automatically implemented by services.

In order for security protocols to be automatically implemented, specifications must also include implementation-related information such as: cryptographic algorithms and modes, cryptographic key sizes, generated random number size, etc. These requirements, added to the information required by the composition, makes the construction of a new specification even more problematic.

By inspecting the technologies available in the field of Web services, we realize that there are several possibilities that can be adapted to comply with the formulated requirements. From these, in our previous work [14], we have chosen WSDL-S and OWL to be the components from which specifications are constructed. Each protocol participant is described by a pair of WSDL-S and OWL files, as shown in figure 1.

The role of the WSDL-S component is to describe the message sequences and directions that must be executed by protocol participants. Each WSDL-S component contains the participant's role in executing the protocol (i.e. initiator, respondent or third-party), protocol preconditions, protocol effects and message sequences.

The role of the OWL component is to provide semantic information such as the construction, processing and implementation of cryptographic operations (e.g. encryption algorithm, encryption mode, key). This is connected to the WSDL-S component via annotations.

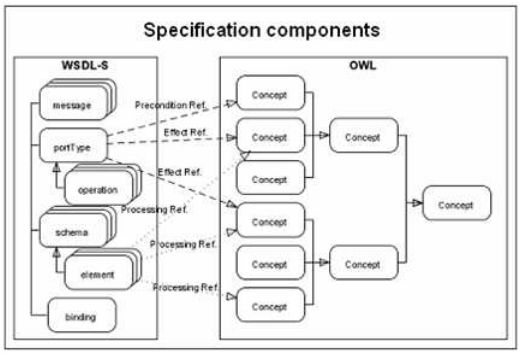


**Figure 1:** Specification components for a protocol participant

## 4 Middleware architecture

### 4.1 Service oriented architecture

Client applications are able to access resources by first locating them, followed by the download of the service and security protocol specifications and by the execution of an authentication sequence. The services provided by the platform must provide a way to publish, locate, compose and automatically access resource services. In addition, in order to face the challenges of rapidly changing protocol specifications, the protocol implementations must provide flexible and extensible components.

Based on these requirements, we define four types of services: name services, specification services, authorization and composition services and resource services. Name services (NAME-S) are implemented through UDDI [5] registries and are used to register, identify or locate existing services. Specifications are stored and managed by specification services (SPEC-S). Specifications are implemented using Web service technologies such as SAWSDL [10], WSDL-S [8] and OWL [9]. Authorization and composition services (AUT-S) implement the verification mechanisms of client credentials and ensure the composition of service capabilities and security protocols. Finally, the resource services (RES-S) implement a set of capabilities provided for client applications.

Accessing resources by a client application is done in several steps, as shown in figure 2. First, the client must establish the set of services implemented by the system (step 1). In order to access a resource or a set of resources, the client must be authenticated, its rights must be verified and, if necessary, resources must be composed. This is done by accessing the AUT-S service, for which the specifications must be first downloaded. The client requests the location of AUT-S and SPEC-S (step 2) from NAME-S, followed by the request of the specifications for AUT-S (step 3). The request for accessing RES-S is sent to AUT-S (step 4), containing the user credentials. These are verified (step 5) and a security token is generated by AUT-S that is sent to RES-S (steps 6, 7, 8). The client receives the generated token and sends it to RES-S (steps 9, 10, 11), after which it is able to access the capabilities provided by the resource.

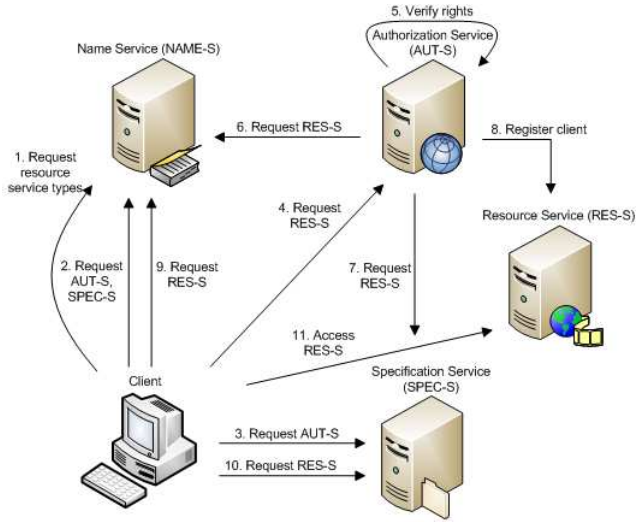


**Figure 2:** Accessing services

### 4.2 Software architecture

The architecture of the software stack is given in figure 3. We identified two main layers: the communication layer and the service layer, given in figure 3.

The *communication layer* provides the implementation of the service and security protocols needed to access service capabilities. It is built on existing network and XML message-based protocols. Security protocols are implemented using extensions of the WS-Security standard, provided in our previous work [13]. The extensions consist of XML constructions for user names and binary keys required by key exchange and authentication protocols. The automated execution of security protocols is based on specifications developed using existing Web service technologies such as WSDL-S [8] and OWL [9], presented in the following sections. Service protocols are described by SAWSDL [10] specifications and are specific to each service. Name

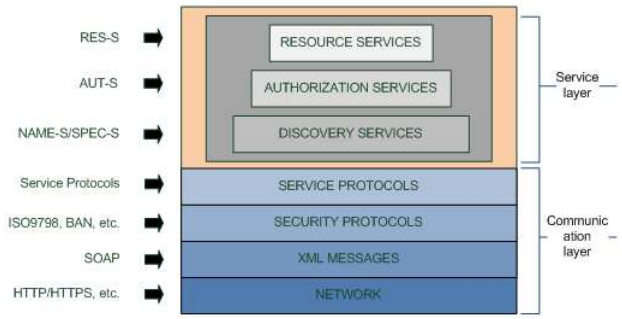


**Figure 3:** Software stack

services define a set of messages for interrogating service registry, while specification services define messages for downloading specifications. Authorization services define messages for requesting access to services and to send security tokens. Resource services provide two types of specifications, for each external entity accessing them. The AUT-S specifications provide messages for setting user security tokens, while the client specifications provide client access messages.

The *service layer* provides the implementation of service capabilities. The capabilities of NAME-S, SPEC-S and AUT-S have already been discussed before. The capabilities of RES-S are specific to each implementation, ranging from video image capabilities to data storage capabilities. In order to implement new resource services, the only components that require change are the capabilities and the service protocol specifications. The security properties of communications with new resources are ensured by the underlying communication layer that remains unchanged.

# 5 A new video surveillance system

## 5.1 Motivation

Over the last decade, because of increasing security concerns, video surveillance systems have received significant attention from both the research and industry communities. The use of surveillance systems has become a necessity in airports, traffic, subways, stores and even homes. The requirements for developing such systems, formulated by Ostheimer et al [15], include affordable hardware requirements, low bandwidth consumption and access control procedures.

The use of the Internet to transfer video images between distributed nodes has made it possible to expand the applicability of these systems to remote surveillance [16], allowing the distribution of nodes across multiple countries and continents. By using the Internet, surveillance systems also adopted modern security protocols such as TLS or VPN to secure data transfers.

In this section we propose a Web service-based approach for implementing video surveillance systems. The proposed system is based on the platform presented in the previous sections and comes with multiple advantages over existing video surveillance systems. First, using existing transport or network layer security in closed environments protected by firewalls or NATs, where access is granted only to HTTP-based protocols becomes a possibility, as opposed to existing systems. Second, the XML-based protocols provide much more flexibility than the binary ones currently used. Third, video surveillance can be applied in systems with heterogenous security protocols, where video services are automatically composed to create more complex services.

## 5.2 System components

The proposed video surveillance system defines three types of resource services: capture services, saving services and replay services. Capture services (i.e. CAP) are used for the actual capturing of video images from connected hardware. These frames can be sent directly to user applications or they can be saved for future visualization. Frames are stored by the saving services (i.e. SAVE) that also deal with the distribution of the captured frames to multiple storage hardware. Saved frames can be viewed by using replay services (i.e. REPL).

To store the captured frames, we apply composition operations on a SAVE service and multiple CAP services. By composing SAVE with CAP services we can define time-based replay synchronization operations for REPL services. Composition operations are handled by the COMP service which composes service capabilities and security protocols used in the communication process.

## 5.3 Experimental results

The experiments we conducted focussed on measuring the time required to access single and composed resources and on the time required to transfer data between services and client applications. For each resource type we used a different station with the same hardware configuration: Intel Dual Core CPU, 1.8GHz, 1GB of RAM, Windows XP OS.

First, we measured the accessing time of single resources (i.e. that do not require composition). As shown in figure 4, operations such as locating and downloading specifications clearly affect system performance. If specifications are not saved, accessing time of single resources is on average 580ms. If specifications are saved, the accessing time drops approx. 200ms. The peaks from the figure denote the

cases when the authorization service also needs to download specifications for the requested services.

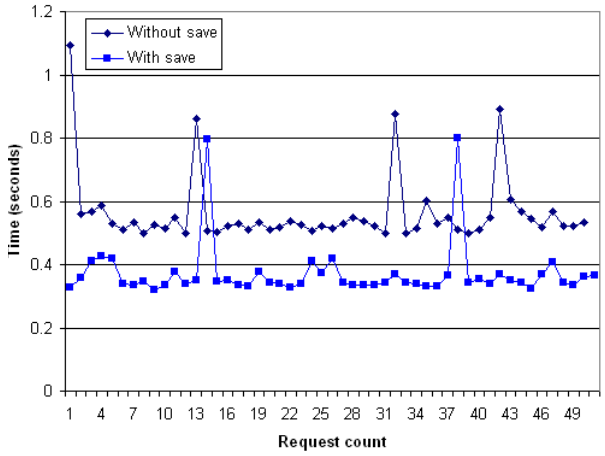


**Figure 4:** Accessing time of single resources

Next, we measured the actual time required for the composition of specifications. Measurement results are shown in figure 5. The performance of the PE-composition and of the PC-composition is relatively good, considering that the composition takes around 450ms for 50 resources. However, the composition time is also influenced by the size and number of specifications, as shown in the same figure. Thus, the actual processing of specifications can reach almost 2.5 seconds for 50 resources.

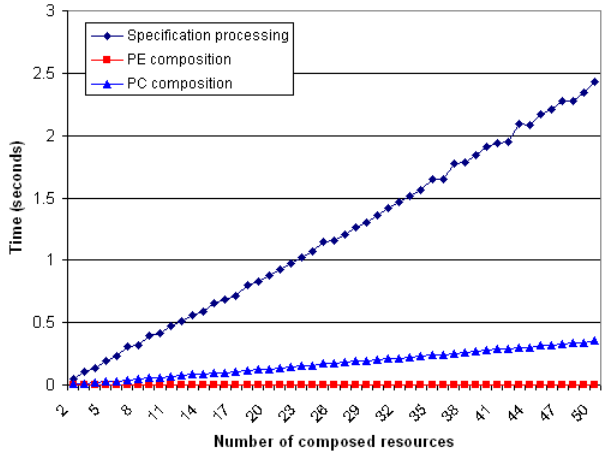


**Figure 5:** Specification composition time

While accessing composed resources, users do not only have to wait for the composition of specifications, but they also have to interrogate multiple services, such as the name service or specification service. Also, in the composition process the authorization service must connect to all resource services involved, must download their specifications and must compose them. These operations all contribute to the total accessing time for composed resources, as shown in figure 6.

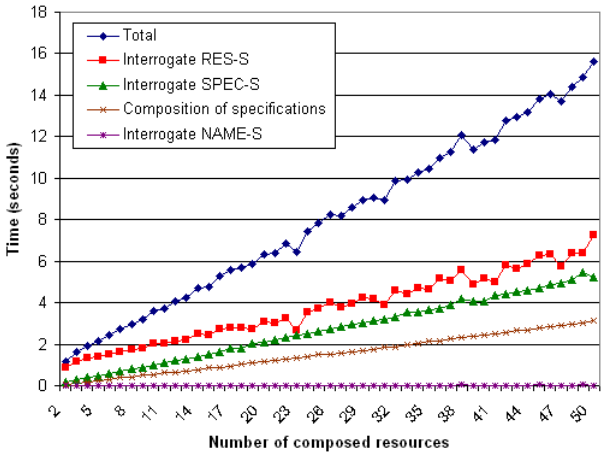


**Figure 6:** Accessing time of composed resources

Another aspect we measured was the time needed for frames to reach client applications in case of composed resources. We considered the two types of resources mentioned at the beginning of this section: SAVE (i.e. SRES-S) and VIDEO (i.e. VRES-S). We composed 2 to 50 VIDEO resources with a single SAVE resource and we measured the average time for frames to reach the SAVE service (i.e. from VRES-S to SRES-S) and the average time for frames to reach the client application from the SAVE resource (i.e. SAVE-Client). Because the transfer time is affected by the size of the frames and by the number of frames/second, we considered two cases. First, we considered a 5fps frame-rate, with the size of 10KB for each frame. For this case, results are shown in figure 7. For another case, we considered a frame-rate of 10fps with the size of 20KB for each frame. Results for this later case are shown in figure 8.

## 6 Conclusions

We presented a new middleware for the automated composition of Web services that user heterogeneous security protocols. The approach used for the composition of security protocols was developed by the authors in their previous work and was implemented in the proposed middleware as a composition module.

The novelty introduced by the paper is the video surveillance system that can be used to create new, composed video resources. In this paper we composed video capture

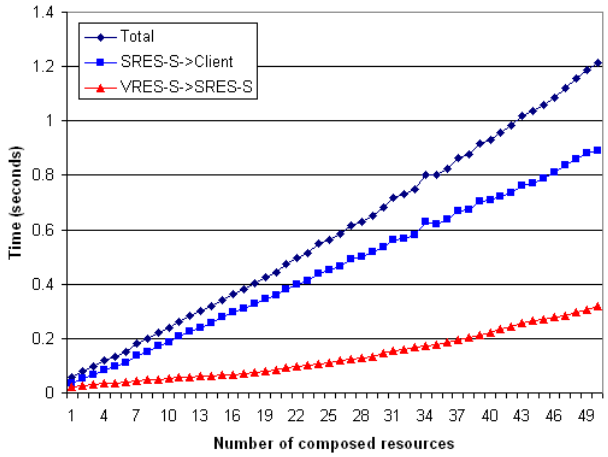


**Figure 7:** Transfer time for 5fps and 10KB/frame

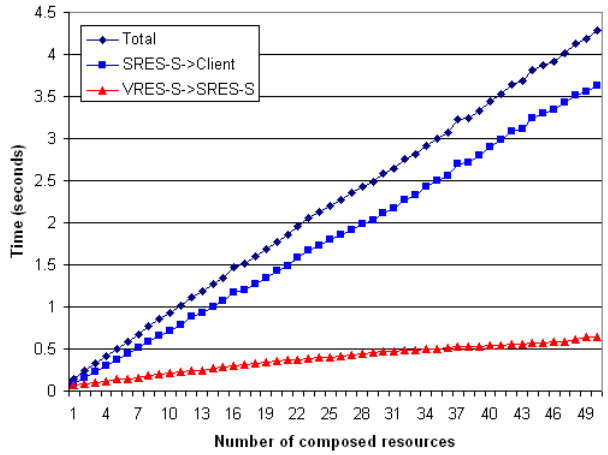


**Figure 8:** Transfer time for 10fps and 20KB/frame

with video save resources in order to provide saving capabilities for frames that must be replayed in the future. The performances of the implemented system are highly dependent on the frame-rate and the size of the capture frames, which can be improved by using dedicated hardware for cryptographic operations or XML parsing.

## References


[1] C.J.F. Cremers, and S. Mauw, "Checking secrecy by means of partial order reduction", In S. Leue and T. Systa, editors, Germany, september 7-12, 2003, revised selected papers LNCS, Vol. 3466, 2005.

[2] B. Genge, I. Ignat, and P. Haller, "Automated Composition of Security Protocols", 2009 IEEE International Conference on Intelligent Communication and Processing, Cluj-Napoca, Romania, pp. 251–258, 2009.

[3] I. B. Arpinar, B. Aleman-Meza, R. Zhang, and A. Maduko, "Ontology-Driven Web Services Composition Platform", Proceedings of the IEEE International Conference on E-Commerce Technology, pp. 146-152, 2004.

[4] R. W. Feenstra, M. Janssen, and R. W. Wagenaar, "Evaluating Web Service Composition Methods: the Need for Including Multi-Actor Elements", The Electronic Journal of e-Government Volume 5 Issue 2, pp. 153 - 164, 2007.

[5] Organization for the Advancement of Structured Information Standards, UDDI, available at http://www.uddi.org/pubs/uddi_v3.htm, 2004.

[6] Organization for the Advancement of Structured Information Standards, Security Assertion Markup Language (SAML), version 2.0, available at http://saml.xml.org/saml-specifications, March 2005.

[7] Organization for the Advancement of Structured Information Standards, Web Service Trust (WS-Trust) version 1.3, available at http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html, March 2007.

[8] World Wide Web Consortium, Web Service Semantics - WSDL-S, W3C Member Submission, 7 November, 2005.

[9] World Wide Web Consortium, OWL Web Ontology Language Reference, W3C Recommendation, 10 February 2004.

[10] Martin, D., Paolucci, M., Wagner, M., "Toward Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspective", OWL-S: Experiences and Directions - workshop at 4th European Semantic Web Conf, 2007.

[11] A. Datta, A. Derek, J.C. Mitchell, and A. Roy, "Protocol Composition Logic (PCL)", Electronic Notes in Theoretical Computer Science, Vol. 172, 1, pp. 311–358, April, 2007,.

[12] S. Andova, C.J.F. Cremers, K. Gjosteen, S. Mauw, S. Mjolsnes, S. Radomirovic, "A framework for compositional verification of security protocols", Special issue on computer security: Foundations and Automated Reasoning, Vol. 206(2-4), Elsevier, pp. 425–459, 2008.

[13] B. Genge, P. Haller, "Extending WS-Security to Implement Security Protocols for Web Services", International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics, Targu Mures, Romania, To appear, 2009.

[14] B. Genge Bela, P. Haller, "Towards Automated Secure Web Service Execution", Networking 2009, Aachen, Germany, May 11-15, Lecture Notes in Computer Science (LNCS 5550), L. Fratta et al. (Eds.), Springer-Verlag, pp. 943–954, 2009.

[15] D. Ostheimer, S. Lemay, D. Mayisela, P. Dagba, M. Ghazal, A. Amer, "A modular distributed video surveillance system over IP", in Proc. IEEE Canadian Conference on Electrical and Computer Engineering, Ottawa, Ontario, Canada, pp. 1001-1004, May 2006.

[16] X. Yuan, Z. Sun, Y. Varol, G. Bebis, "A distributed visual surveillance system", IEEE international conference on advanced video and signal based surveillance proceedings, pp. 199–204, July 2003.