Distributed Privacy for Visual Sensor Networks via Markov Shares

William Luh and Deepa Kundur
Texas A&M University
Department of Electrical and Computer Engineering
214 Zachry Engineering Center, TAMU 3128, College Station, TX 77843
Email: {luh, deepa}@ece.tamu.edu

Abstract

Visual sensor networks (VSNs) can be used to acquire visual data (i.e. images) for applications such as military reconnaissance, surveillance, and monitoring. In these applications, it is of utmost importance that visual data be protected against eavesdropping to uphold confidentiality and privacy rights. Furthermore, protection mechanisms for these sensor nodes must be efficient and robust to node capture and tampering. This paper considers a distributed approach to privacy in which highly correlated images within a dense sensor cluster are obfuscated. The particular approach, in which nodes within a cluster work together to create and transmit shares (called Markov shares) makes it necessary for an attacker to capture several correlated visual nodes and/or shares in order to gain improved semantic information of the observation area. The proposed technique does not require that the individual sensor node readings be exactly registered, nor the correlation model be known a priori. Simulation results based on a cluster of 18 *nodes show: (1) most Markov shares use fewer bits per pixel than the original image hence providing compression capability; (2) a denial of service attack on a single node (e.g., corrupting a region of interest) has minimal impact on the reconstructed data at the sink; (3) five or more Markov shares need to be intercepted by an attacker before the semantic content of the desired image can be understood; (4) authorized reconstruction of unregistered individual images with random rotation transformations up to* 10 *degrees is possible.*

1 Introduction

Visual sensor networks (VSNs) are a form of distributed networked sensors in which a subset of the nodes collect visual data. Such perceptually rich data is more accessible to human *sinks* for greater interactivity. For applications including healthcare surveillance, environmental monitoring, and vehicle control, visual data provides crucial signatures for monitoring. In addition, VSNs are a convenient framework to interface emerging scalar wireless sensor networks (WSNs) with existing (wired or wireless) video surveillance infrastructure. The proliferation of low-cost portable off-the-shelf media sensing devices has motivated the recent development of VSN architectures, systems, and testbeds.

Widespread public adoption of VSNs rests on the ability of developers to address privacy concerns. The recent domestic eavesdropping program has resuscitated the call for privacy rights. Furthermore, given the use of video footage by authorities in the recent London bombings, it is likely that such information systems will become a target of future terrorist attacks. It is, therefore, imperative that issues of VSN protection be addressed. Previous techniques for sensor network security implicitly assume low-bandwidth scalar sensor readings and RF communications and are not directly applicable to VSNs for the following reasons: (a) the value of raw scalar sensor readings is often considered to be much lower than the value of raw visual data in VSNs which require privacy protection, (b) the high bandwidth of visual data requires protection mechanisms that are lightweight to be suitable for sensor networks, (c) visual data protection often has different objectives for privacy; for example *semantic* or *subject* privacy goals require that a selected portion of the content be confidential in comparison to scalar data in which often the entire stream needs to be secured. Techniques for security of traditional visual surveillance systems are also not applicable because: (a) trusted-third-party (TTP) infrastructure may not be available due to the highly distributed architecture of VSNs, (b) the high probability of physical compromise of some nodes makes it necessary to incorporate mechanisms that can distill the impact of node capture, as well as overcome insider attacks at all hierarchical levels in the VSN, (c) wireless communications and battery/solar powered visual nodes are likely limiting computational and communication capability at the nodes.

The unique characteristics of VSNs necessitate new se-

Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)
0-7695-2529-6/06 $20.00 © 2006 IEEE

curity paradigms that effectively exploit their inherent characteristics. In this paper, we consider an approach to providing *distributed privacy* that is more robust to node compromise and insider attack that leverages visual *obfuscation* to provide confidentiality. Distributing trust is an approach that has shown great promise for security in scalar sensor networks [4] and obfuscation, which is the art of complicating data semantics without changing functionality has received attention for software and multimedia protection as a promising lightweight security mechanism [1]. We demonstrate within the context of visual secret sharing how distributed trust and obfuscation of content lends itself to a solution for privacy in VSNs. To the best of the authors' knowledge, this is the first contribution that looks at distributed privacy in VSNs.

In the following subsections, we describe the general problem setup along with assumptions and constraints on our VSN architecture. We then outline existing work that has inspired the proposed solution, and point out necessary extensions offered by our approach.

1.1 General

The general VSN set-up is summarized in Figure 1. N densely deployed sensors equipped with digital cameras (with image and/or video acquisition facility) and limited video processing capabilities are in close proximity to one another [6] such that they can record the same scene, but from slightly different perspectives. These nodes are said to form a *cluster*. These N sensors cooperate (with limited communications) to compress and obfuscate the images they acquire so that they are unintelligible. The resulting compressed and "secured" images (see Section 1.2 for the specific security requirements), which we call *shares*, are then sent to M (preferably close to N) distributed storage stations, which may be in close proximity to the visual node (e.g., attached to it), in the same building, or significantly physically separated.

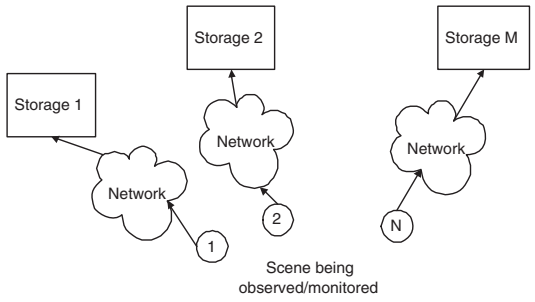

Figure 1. General setup

We assume that there is a common image scene, called the *representative image*, that is acquired by each of the N sensors. To distribute trust, obfuscation occurs such that each of the sensor outputs is unintelligible and only when approximately all N distinct shares are available, can the representative image be reconstructed by the sink (who has access to all M storage locations). Figure 2 shows the general reconstruction process. For an attacker to get significant information of the common scene viewed by the visual sensors, (s)he would have to intercept a large number of shares while in transit to the storage stations or at their distributed storage locations. This raises the level of effort required to overcome privacy and increases the robustness to insider attacks which would require that a large number of nodes be compromised without knowledge to the network administrator.

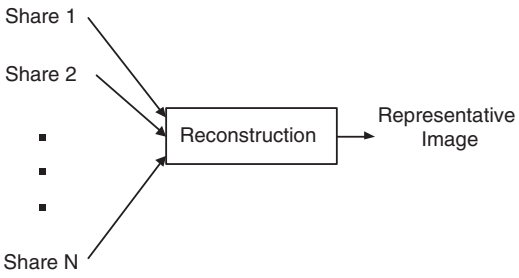

Figure 2. Reconstruction

We outline other goals that a solution should strive for:

(1) Compression is performed at each sensor node using side information communicated from only one other node in the cluster.

(2) Each sensor node within the cluster obfuscates its image using side information from only one other node in the cluster. This requirement should be achieved jointly with the requirement above for efficiency.

(3) Distinct keying information known only to each node is distributed across the sensors within a cluster, so that a capture of one node will not reveal all the keys (i.e. parameters) necessary for reconstruction.

(4) Keying information is distributed across the M storage stations, to hinder insider attacks.

Minimizing the number of bits transmitted in Goal 1 is necessary in any sensor network, in particular VSNs, to minimize bandwidth and storage usage. Goal 2 provides security against eavesdropping. Goal 3 protects the network against node capture, since distributing key information across several nodes will minimize the impact of a single node capture.

Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)
0-7695-2529-6/06 $20.00 © 2006 IEEE

1.2 Security requirements and assumptions

In generating the N shares, each share must satisfy the following security requirements:

(1) Each share does not fully reveal, visually, the original image. Some minor details are permitted [12].

(2) Given only one share, and any parameters used to create the share, an attacker cannot derive a visual approximation (see Requirement 1) of the original image without heavy computational complexity.

(3) If a region of interest of one share is maliciously corrupted, the other $N-1$ shares should still be able to visually recover the corrupted region of interest. In other words, a local corruption of data in one node should have minimal impact on the reconstructed data at the sink.

Also, given a threshold T, an attacker with T or fewer shares cannot derive a good visual approximation of the representative image. Requirements 1 and 2 deal with the privacy/confidentiality problem, while Requirement 3 deals with the different problem of tampering. Next we impose limitations on the attacks:

(1) The attacker can only intercept a small number of shares, and capture a small number of nodes.

(2) Node capture is defined in this paper to be the physical removal of a node from its cluster. Once a node is removed, it cannot rejoin the VSN. The VSN is also able to re-organize its logistics to cope without the missing node.

(3) The attacker cannot electronically wiretap a node, meaning, (s)he cannot internally eavesdrop on a node's internal memory and processor from some distance.

Constraint 2 is to avoid an attacker capturing a node and all its keying information, and then throwing it back into the cluster. This kind of attack is best left to *sensor network key management* [5]. Constraint 3 may seem stringent, however if an attacker could eavesdrop internally, then (s)he could essentially intercept the representative image before it is obfuscated.

To the best of the authors' knowledge, no existing solution fulfills all the above (Sections 1.1 and 1.2) requirements.

1.3 Previous work

In this section, which is not intended to be an exhaustive survey, we outline the body of existing work that inspired our approach, and point out where we extend them. We emphasize that a solution to our problem is meant to compliment existing work and vice versa. For example, traditional encryption techniques can further be applied to each of the N shares to force attackers to not only acquire almost all N shares, but in addition require an attacker to capture all key information as well.

Secret sharing [13] is a cryptographic technique, in which a trusted third party, called the *dealer*, encrypts a plaintext with a key K, hence producing a ciphertext. The dealer then distributes N keys $\{K_i\}_{i=1}^N$, called *shares* to N users, called *participants*, such that certain subsets $A \subseteq \{K_i\}_{i=1}^N$ can be used to reconstruct the key used for decryption; this key is K for symmetric key cryptography. Unauthorized subsets cannot be used to reconstruct the decryption key. For example, the secret sharing algorithm can require that all N shares be available to reconstruct the decryption key. Hence if any one, two, or fewer than N shares are captured, the ciphertext cannot be decrypted. This is essentially what we require of a solution to our distributed privacy problem. However, traditional secret sharing requires a TTP, the dealer, who not only creates the shares, but actually encrypts the plaintext. In our VSN, each node creates its own share, hence traditional secret sharing as is, does not suffice to solve our problem.

In visual secret sharing (VSS) [10, 2, 7, 8], the same TTP architecture is used, except that each participant receives an unintelligible *shadow image*, such that when a subset of shadow images is combined (such as OR for binary images) without the use of a key, the original image is approximately reconstructed. In VSS, the dealer creates shadow images via a homophonic substitution cipher [10], and hence the homophonic function needs to be kept safe from attackers. Unfortunately this homophonic function has the same importance as a key, and is susceptible to capture under our hostile assumption. Furthermore, early VSS schemes create shadow images larger in size than the original image, and hence opposes compression.

We therefore consider a secret sharing methodology based on a discrete-time Markov chain system. This approach detailed in the next section provides a method that makes it feasible to have a distributed obfuscation method which is naturally robust to model mismatch (e.g., for unregistered data), is easy to implement because it is based on well-founded theory, and provides semantic or subject privacy with compression because perceptual and energy constraints can be simultaneously incorporated into the formulation. The keys or homophonic functions in secret sharing and VSS that are traditionally located at the TTP can now be distributed across the VSN, hence having the ability to distill the impact of a single node capture.

Finally, we point out that a coding approach to our problem, such as the use of symmetric rate Slepian-Wolf codes

[14] is plausible. In this paper, we present a signal processing approach to our problem, and show that our approach does not require *a priori* knowledge of the sensor correlation model, nor the attack model to some extent. This is in contrast to a coding approach, for example using syndromes for Slepian-Wolf coding [11], in which a sensor correlation model and attack model are required; often it is difficult to predict the attack model. We are currently researching a solution which combines coding and signal processing to provide a tradeoff between robustness and efficiency.

2 Secret sharing using Markov shares

In this section, we first motivate our approach, then present the general idea behind our approach, and finally provide security and implementation insights.

2.1 Motivation

As mentioned in Section 1.3, traditional secret sharing schemes have a centralized architecture in which a TTP generates the shares and transmits these shares to N nodes. The TTP is a single point of vulnerability to attacks, and hence we strive to distribute the approach to distill this vulnerability. We see from Figure 3 that N communication arrows fan-out from the TTP in a parallel manner in classical secret sharing. It is then natural that to achieve the kind of distribution required, the communication arrows could be of a more serial structure as seen on the right side of Figure 3. Our method is based on this *chain* structure, which lends itself to algorithms that make use of the state equations of dynamical systems. Systems described by such state equations are privileged with well-known control methods that are robust to noise, for example due to lack of registration or denial of service attacks, which are of interest in our problem. This is the motivation behind adopting the method described in the next section.

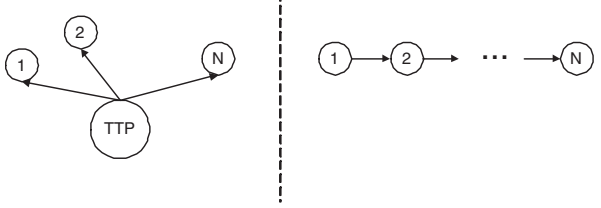

Figure 3. Classical secret sharing architecture (parallel communication arrows) vs. distributed architecture (serial communication arrows)

2.2 General idea

First we assume that all our sensors capture identical grayscale images[1], which we write as a column-vector, $\overline{\mathbf{x}}$ (after having performed a column-wise raster scan of its image matrix), and that there exists a central authority that creates the N shares. Essentially, $\overline{\mathbf{x}}$ can be perceived as our representative image. The central authority creates shares through controlling (regulating) a discrete-time Markov chain system, Σ_{Markov}, described by the state equation in Equation 1.

$$\Sigma_{\text{Markov}}: \quad \mathbf{x}_{k+1} = f_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \tag{1}$$

Let $\mathbf{x}_0$ represent the initial state of the system Σ_{Markov}, then $\mathbf{x}_k$ is the state of Σ_{Markov} at time k. Also Σ_{Markov} takes an external input/control at each time instance, so that $\mathbf{u}_k$ is the input/control at time k. $\mathbf{w}_k$ is some random perturbation drawn from a fixed distribution, which can account for a lack of synchronization of the images captured by different sensors, as well as from attacks. The central authority designs f_k *a priori* so that it is controllable, and so the security requirements in Section 1.2 are satisfied. Figure 4 shows the states of Σ_{Markov} evolve over time. Suppose we wish to drive Σ_{Markov} from the initial state $\mathbf{x}_0$ (drawn independently of $\overline{\mathbf{x}}$), to the image $\overline{\mathbf{x}}$; that is, we wish $\mathbf{x}_N \approx \overline{\mathbf{x}}$ in Figure 4. In doing so, if the central authority distributes the N controls $\{\mathbf{u}_i\}_{i=0}^{N-1}$, and discards all intermediate states x_i, $1 \leq i \leq N$, then these N controls can be used as the N shares, provided that they satisfy the security requirements in Section 1.2. We call these shares, *Markov Shares* (or *MarS*), since they are generated from Σ_{Markov}.

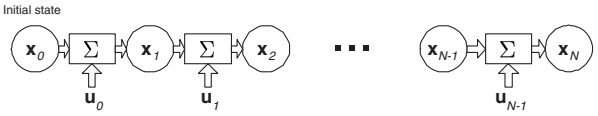

Figure 4. Discrete-time Markov chain system

To reconstruct the representative image $\overline{\mathbf{x}}$, we note that $\mathbf{x}_N \approx \overline{\mathbf{x}}$. Hence given the initial state $\mathbf{x}_0$, and MarS, $\{\mathbf{u}_i\}_{i=0}^{N-1}$, one can retrace the state evolution from $\mathbf{x}_0$ to $\mathbf{x}_N \approx \overline{\mathbf{x}}$, thereby obtaining an approximation of the representative image.

We briefly describe the security, and defer the details to Section 3.2, where we study the security for a specific system f_k. If an attacker is able to intercept one MarS, $\mathbf{u}_i$, retracing the state evolution is not possible if Σ_{Markov} is not fully known and the intermediate states are also not known.

[1] We relax the identical image assumption later in the simulation results.

Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)
0-7695-2529-6/06 $20.00 © 2006 IEEE

2.3 Generation of Markov shares

We now describe the procedure behind generating MarS. In order to quantify $\mathbf{x}_N \approx \overline{\mathbf{x}}$, we define $d(\mathbf{x}_i, \overline{\mathbf{x}})$ to be a cost metric measuring the distance between state $\mathbf{x}_i$ and representative image $\overline{\mathbf{x}}$. In order to collectively minimize the number of bits of the MarS, we define $\mathcal{E}(\mathbf{u}_i) > 0$ to measure the *size* of each $\mathbf{u}_i$. Generation of MarS is described by the optimization problem in Equation 2.

$$\arg\min_{\{\mathbf{u}_i\}} E\left\{ d(\mathbf{x}_N, \overline{\mathbf{x}}) + \sum_{k=0}^{N-1} \{d(\mathbf{x}_k, \overline{\mathbf{x}}) + \mathcal{E}(\mathbf{u}_k)\} \right\} \quad (2)$$

In Equation 2, the expectation E is over the random variables $\mathbf{w}_i$. In addition to $\mathbf{x}_N \approx \overline{\mathbf{x}}$, we wish the collective size of all the MarS to be minimized, which is accounted for by $\mathcal{E}(\mathbf{u}_k)$ inside the sum. We note that $d(\mathbf{x}_N, \overline{\mathbf{x}})$ is not inside the sum, because there is no matching $\mathbf{u}_N$; the last MarS is $\mathbf{u}_{N-1}$. We also note that $\mathcal{E}(\cdot)$ should be designed appropriately to prevent $\mathbf{x}_i$ from converging to $\mathbf{x}_N$ too soon; this scenario would cause the remainder MarS to be $\mathbf{0}$. The solution to Equation 2 under the Markov assumption can be derived using dynamic programming [3]. The complexity of the dynamic programming solution depends on the Σ_{Markov} as well as the cost metrics.

2.4 Distributing Markov share generation

The formulations above require a central authority to generate the MarS. We now discuss how it is possible to fulfill the distributed requirements mentioned in Section 1.1. We note that our approach is geared towards visual data such as natural images, in which large noise variations can be tolerated without completely compromising the semantics of the data. Again we assume for the time-being that all sensor nodes are capturing the representative image $\overline{\mathbf{x}}$, i.e. they are all capturing the identical image.

Suppose each of the N nodes is associated with a node number 0 to $N-1$. Let the initial state $\mathbf{x}_0$ be stored on Node 0. Let Node 0 have the necessary parameters required to create $\mathbf{u}_0$ and hence also create $\mathbf{x}_1$. In doing so, Node 0 then encrypts $\mathbf{x}_1$ using a symmetric key it shares with Node 1, then transmits the encrypted $\mathbf{x}_1$ to Node 1, upon which $\mathbf{x}_1$ is removed from Node 0's memory . Node 1 will decrypt the encrypted $\mathbf{x}_1$, and then use $\mathbf{x}_1$ and any necessary parameters to create $\mathbf{u}_1$, and hence also create $\mathbf{x}_2$, which is similarly encrypted and sent to Node 2, upon which $\mathbf{x}_1$ and $\mathbf{x}_2$ are removed from Node 1's memory. In general, Node k will share a unique symmetric key with Node $k+1$ for $0 \le k \le N-2$; for any Node i, $1 \le i \le N-2$, Node i:

(1) receives from Node $i-1$, the encrypted $\mathbf{x}_i$, which is decrypted and used to create its share $\mathbf{u}_i$;

(2) creates $\mathbf{x}_{i+1}$ from Equation 1, setting $\mathbf{w}_k = \mathbf{0}$;

(3) sends encrypted $\mathbf{x}_{i+1}$ to Node $i+1$;

(4) removes the states $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ from memory if they are still present.

For the last node, Node $N-1$, only Steps 1 and 4 above are applied. We note that Step 4 is required because the security is intrinsically tied to the attacker not having the intermediate states, hence it is vital that these states are removed from a node's memory in case of node capture.

2.5 Security primitives

First, we give *our* definition of a capture attack in Definition 1.

Definition 1 (Capture Attack) *A capture attack is one in which an attacker obtains all necessary keys or secret parameters through node capture, and solves for $\overline{\mathbf{x}}$.*

We show that if the attacker has captured a specific subset of nodes, and further intercepts exactly one share, then the attacker can solve for $\overline{\mathbf{x}}$ simply by solving one equation with one unknown.

Proposition 1 *Let each share be created as in Equation 3.*

$$\mathbf{u}_i = \mu_i(\mathbf{x}_i, \overline{\mathbf{x}}) \quad (3)$$

We assume that given $\mathbf{u}_i$ and $\mathbf{x}_i$, we may solve for (i.e. compute) a unique $\overline{\mathbf{x}}$. If the attacker knows $\mathbf{x}_0$, $\{\mu_0, \ldots, \mu_i\}$, $\{f_0, \ldots, f_{i-1}\}$, and $\mathbf{u}_i$, then (s)he may derive $\overline{\mathbf{x}}$, for any $1 \le i \le N-1$.

For the trivial case when $i = 0$, we have $\mathbf{u}_0 = \mu_0(\mathbf{x}_0, \overline{\mathbf{x}})$, and since the attacker knows $\mathbf{x}_0$, μ_0, and $\mathbf{u}_0$, (s)he may solve for $\overline{\mathbf{x}}$. We use induction to show that Proposition 1 is true for any $1 \le i \le N-1$.

Proof 1 *Consider $i = 1$ as the base case in Equation 4.*

$$\mathbf{u}_1 = \mu_1(\mathbf{x}_1, \overline{\mathbf{x}}) \quad (4)$$

Since we can write $\mathbf{x}_1 = f_0(\mathbf{x}_0, \mu_0(\mathbf{x}_0, \overline{\mathbf{x}}))$, we can substitute this back into Equation 4. Now since the attacker knows $\mathbf{x}_0$, $\{\mu_0, \mu_1\}$, f_0, and $\mathbf{u}_1$, (s)he may solve for $\overline{\mathbf{x}}$ the remaining unknown in the associated equation.[2] *So the case $i = 1$ is true. Let the case $i-1$ be assumed to be true as our induction hypothesis. This implies that the attacker may solve for $\overline{\mathbf{x}}$ given $\mathbf{x}_0$, $\{\mu_0, \ldots, \mu_{i-1}\}$, $\{f_0, \ldots, f_{i-2}\}$, and $\mathbf{u}_{i-1}$ in Equation 5.*

$$\mathbf{u}_{i-1} = \mu_{i-1}(f_{i-2}(\mathbf{x}_{i-2}, \mu_{i-2}(\mathbf{x}_{i-2}, \overline{\mathbf{x}})), \overline{\mathbf{x}}) \quad (5)$$

[2] In this paper, we assume the best case scenario for the attacker in which the attacker may solve an equation with a single unknown.

Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)
0-7695-2529-6/06 $20.00 © 2006 IEEE

We now prove the induction step for the case i. Here the attacker knows $\mathbf{x}_0$, $\{\mu_0, \ldots, \mu_i\}$, $\{f_0, \ldots, f_{i-1}\}$, and $\mathbf{u}_i$. We can expand $\mathbf{u}_i$ as follows.

$$\begin{aligned} \mathbf{u}_i &= \mu_i(\mathbf{x}_i, \overline{\mathbf{x}}) && (6)\\ &= \mu_i(f_{i-1}(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}), \overline{\mathbf{x}}) && (7)\\ &= \mu_i(f_{i-1}(\mathbf{x}_{i-1}, \mu_{i-1}(\mathbf{x}_{i-1}, \overline{\mathbf{x}})), \overline{\mathbf{x}}) && (8) \end{aligned}$$

If we substitute $\mathbf{x}_{i-1} = f_{i-2}(\mathbf{x}_{i-2}, \mu_{i-2}(\mathbf{x}_{i-2}, \overline{\mathbf{x}}))$ into Equation 8, then the attacker may solve for $\overline{\mathbf{x}}$, since $\mathbf{x}_{i-2}$ is not a function of $\mathbf{u}_{i-1}$ and by the induction hypothesis the term $f_{i-2}(\mathbf{x}_{i-2}, \mu_{i-2}(\mathbf{x}_{i-2}, \overline{\mathbf{x}}))$ could be used to solve Equation 5. ■

In terms of distributing the Markov share generation process, Proposition 1 tells us that if we create shares using μ_i as in Equation 3, then we must ensure that μ_i cannot reveal $\mathbf{x}_0$, $\{\mu_0, \ldots, \mu_{i-1}\}$, $\{f_0, \ldots, f_{i-1}\}$. If μ_i does reveal these functions, then an attacker who intercepts $\mathbf{u}_i$ and captures the corresponding Node i, thus obtaining μ_i stored on-board, may use Proposition 1 to solve for $\overline{\mathbf{x}}$.

Definition 2 (Unrevealing Share Generating Function) *μ_i, a function generating share i is said to be an unrevealing share generation function if μ_i does not reveal $\{\mu_0, \ldots, \mu_{i-1}\}$, $\{f_0, \ldots, f_1\}$.*

In Definition 2, we do not stipulate that μ_i does not reveal $\mathbf{x}_0$, because most μ_i do not do so anyway. $\mathbf{x}_0$ is usually stored exclusively in Node 0. Also, we note that unrevealing share generating functions are necessary for security, but not sufficient; that is implementing unrevealing share generating functions does not guarantee security from all possible capture attacks, but at least it will prevent the specific attack in Proposition 1.

Creating a bad μ_i essentially means we are not *spreading* key information across the nodes in a cluster as required in Goal 3 in Section 1.1. Finally we see that if an attacker intercepts $\mathbf{u}_0$ and captures Node 0, then an attacker can always solve for $\overline{\mathbf{x}}$ when μ_0 is as in Equation 3. In Section 3.2, we propose discarding (i.e. never transmitting) $\mathbf{u}_0$ to avoid making Node 0 a single point of failure, and in the simulation results, we show that reconstruction is not adversely affected by this missing share.

3 Example system

In this section, we demonstrate an example of Σ_{Markov}, f_k, and cost metrics, which is also used in the simulation results. We note that there is a large class of Σ_{Markov}, f_k, as well as cost metrics for which our approach can be applied. Suppose Σ_{Markov} is given in Equation 9.

$$\Sigma_{\text{Markov}}: \quad \mathbf{x}_{k+1} = A_k\mathbf{x}_k + \mathbf{u}_k + \mathbf{w}_k \qquad (9)$$

Here A_k is a matrix of appropriate size, and therefore f_k is linear. Σ_{Markov} can be controlled if the pair $(A_k, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix of the same dimensions as A_k, is controllable.

Next, we define our metric $d(\cdot,\cdot)$ in Equation 10.

$$d(\mathbf{x}_i, \overline{\mathbf{x}}) = (\mathbf{x}_i - \overline{\mathbf{x}})^T Q_i (\mathbf{x}_i - \overline{\mathbf{x}}) \qquad (10)$$

Here Q_i is positive semi-definite symmetric, so that $d(\cdot,\cdot) \geq 0$. Finally, we define $\mathcal{E}(\cdot)$ to measure the size of each MarS in Equation 11.

$$\mathcal{E}(\mathbf{u}_i) = \mathbf{u}_i^T R_i \mathbf{u}_i \qquad (11)$$

Here, R_i is positive definite symmetric. Equation 11 essentially measures the energy of $\mathbf{u}_i$. This is somewhat proportional to the number of bits that will eventually encode $\mathbf{u}_i$, because minimizing the energy effectively minimizes the dynamic range of values that $\mathbf{u}_i$ can take on. For example, a number b will require more bits to encode than a number a, when $|b| \gg |a|$, and both are equally probable. Our optimization problem in Equation 2 now becomes a quadratic cost optimization problem in Equation 12.

$$\begin{aligned} \arg\min_{\{\mathbf{u}_i\}} E\Big\{ \quad & (\mathbf{x}_N - \overline{\mathbf{x}})^T Q_i (\mathbf{x}_N - \overline{\mathbf{x}}) + \\ & \sum_{k=0}^{N-1} \Big\{ (\mathbf{x}_k - \overline{\mathbf{x}})^T Q_k (\mathbf{x}_k - \overline{\mathbf{x}}) + \\ & \mathbf{u}_k^T R_k \mathbf{u}_k \Big\} \Big\} \end{aligned} \qquad (12)$$

The reason for our choice of the system and cost metrics is because the solution to the optimization problem of Equation 12 given Σ_{Markov} of Equation 9 is well-known from dynamic programming, and has a closed-form expression [3]. Equations 13 to 17 show the solution to Equation 12.

$$\begin{aligned} \mathbf{u}_k &= L_k(\mathbf{x}_k - \Lambda_k \overline{\mathbf{x}}) && (13)\\ \Lambda_k &= A_k^{-1} A_{k+1}^{-1} \cdots A_{N-1}^{-1} && (14)\\ L_k &= -(R_k + K_{k+1})^{-1} K_{k+1} A_k && (15)\\ K_N &= Q_N && (16)\\ K_k &= A_k^T (K_{k+1} - K_{k+1}(K_{k+1} \\ &\quad + R_k)^{-1} K_{k+1}) A_k + Q_k && (17) \end{aligned}$$

Looking at Equation 13, we see that each MarS is a function of the representative image, and one state - the same form given by Equation 3.

3.1 Implementation insights

For the particular system and cost of our example in Section 3, we address the implementation issues. First we

Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)
0-7695-2529-6/06 $20.00 © 2006 IEEE

note, that each Node k will have L_k and Λ_k stored on the sensor node. These matrices are computed ahead of time, and hence the sensor nodes do not concern themselves with Equations 14 to 17. For our simulations, we choose the matrices A_k to be diagonal, and have non-zero diagonal elements. It is easy to see that this ensures that the pairs $(A_k, \mathbf{I})$ are controllable. In addition, we choose the diagonal entries from the open interval $(1, 2)$ for convergence reasons. The matrices Q_k and R_k are also chosen to be diagonal matrices with non-zero positive diagonal elements. This ensures that they are positive semi-definite and positive definite symmetric respectively. In our simulations, we set all Q_k to the identity matrix, and we vary R_k logarithmically over k to control the size of $\mathbf{u}_k$ over k. In addition, in choosing our matrices to be diagonal, matrix multiplication and inverse degenerates to that of element-wise multiplication and division, which are computationally feasible for sensor nodes with image processing capabilities; essentially processing is performed on a pixel-by-pixel basis.

Also, since these matrices are constants, the diagonal matrices of Equation 13 can be stored as constant vectors (i.e. the diagonals) in the nodes.

3.2 Security insights

We need a specific system and cost defined to study our proposed scheme, and hence in this section we will study the security specific to the example in Section 3. In particular, we look at possible attacks to our example. We consider two types of attacks. In *capture attacks* (Definition 1), the goal of the attacker is to capture enough keys or secret parameters in order to *solve for* $\overline{\mathbf{x}}$. Proposition 1 discussed this type of attack. In *heuristic attacks*, an attacker does not have enough keys or secret parameters, and will estimate these parameters or apply any other means to *approximate* $\overline{\mathbf{x}}$.

3.2.1 Capture attacks

An attacker who intercepts the share $\mathbf{u}_0$ and captures Node 0 to obtain $\mathbf{x}_0$, L_0, and Λ_0 can solve for $\overline{\mathbf{x}} = \Lambda_0^{-1}(\mathbf{x}_0 - L_0^{-1}\mathbf{u}_0)$. This means that we cannot transmit MarS $\mathbf{u}_0$ without compromising security. We will see in the simulation results that reconstruction is still possible upon discarding $\mathbf{u}_0$.

In general, when an attacker captures a node, and intercepts that node's corresponding share, the attacker will have $\mathbf{u}_k$, L_k and Λ_k (the latter two define μ_k of Equation 3). From Proposition 1, we want to check if μ_k reveals $\mathbf{x}_0$, $\{\mu_0, \ldots, \mu_{k-1}\}$, and $\{f_0, \ldots, f_{k-1}\}$.

Proposition 2 *Let $\mu_k(\mathbf{x}_k, \overline{\mathbf{x}}) = L_k(\mathbf{x}_k - \Lambda_k\overline{\mathbf{x}})$, where L_k and Λ_k are defined by Equations 14 to 17, then μ_k for $k \neq 0$ is an unrevealing share generating function.*

Proof 2 *Λ_k is a function of $A_k, A_{k+1}, \ldots, A_{N-1}$. It is easy to see that L_k is also a function of $A_k, A_{k+1}, \ldots, A_{N-1}$. Hence to generate μ_{k-1} given μ_k, one also needs A_{k-1}, which is not present in Λ_k and L_k. In fact, given μ_k, it will be impossible to generate all $\mu_{k-1}, \mu_{k-2}, \ldots \mu_0$ via the same arguments. Now f_{k-1} is a function of A_{k-1}, which is again not present in Λ_k and L_k. By the same arguments, μ_k cannot generate all $f_{k-1}, f_{k-2}, \ldots, f_0$.* ■

The proof above suggests that if the attacker aims for shares and nodes close to Node 0, (s)he may need to capture less nodes. In the next section, we show a heuristic to counter attacks of this form (i.e. capturing nodes close to Node 0), which also takes care of a specific and related heuristic attack.

3.2.2 Heuristic attacks

If we look at $\mathbf{u}_1 = L_1(\mathbf{x}_1 - \Lambda_1\overline{\mathbf{x}})$, we see that although an attacker does not have $\mathbf{x}_1$, an attacker might be able to replace $\mathbf{x}_1$ in this equation with $\mathbf{x}_0$ if the attacker intercepts share $\mathbf{u}_1$, and captures both Node 0 and Node 1, to solve for a visual approximation of $\overline{\mathbf{x}}$. Our simulation results show that if we add some small noise to $\mathbf{u}_1$ (i.e. $\mathbf{u}_1 + \mathbf{w}_1$ in Equation 9), the attacker's visual approximation will suffer. Reconstruction does not suffer as much, because we can use the other MarS to compensate.

Adding noise to obfuscate or even encrypt an object is not new. The one-time pad [15] essentially adds noise/error to a plaintext, in which authorized personnel can remove because they know the exact noise/error added. Shamir's threshold secret sharing [13] can be thought of as each share being created by adding controlled noise to the secret, such that when enough shares are available, one can solve for the secret, hence removing the noise. Finally, the McEliece public-key cryptosystem [9] also adds errors to an error-control-coded plaintext, such that only authorized personnel know the specific error control code used, and hence can remove the errors. In our case, the use of a control-theoretic approach provides much wanted robustness against noise/tampering when a large number shares are available. We exploit this for additional security by perturbing the individual shares with random noise in order to prevent an attacker, who has a small subset of shares and/or captured nodes from gaining semantic visual data. The robustness feature, which holds when a large number of shares are available for reconstruction, does not apply to such practical attacks providing an additional form of protection.

This is but one heuristic attack. In general, all heuristic attacks cannot be predicted in advance, and hence we cannot and do not claim that our scheme is secure under all attacks. We remind the reader that these security issues are specific to the linear system and quadratic cost used in our example. Different choices of system and cost metrics

will result in different security issues, some offering better security than that of our example.

4 Simulation results

In this section we present the simulation results based on the example system and cost of Section 3. We choose $N = 20$, which is an appropriate number of nodes for a dense cluster capturing highly correlated images. In Figure 5(c),

Figure 5. (a) Original representative image (© come.to/torontobus); (b) Markov share $\mathbf{u}_9$; (c) 20 Markov shares

the 4 MarS $\{\mathbf{u}_i\}_{i=1}^{4}$ have 0-mean, 0.05-variance Gaussian noise added to inhibit attacks on these 4 MarS (see security reasons in Section 3.2).

We see that the last MarS $\mathbf{u}_{19}$ in Figure 5(c) cannot be transmitted, because it reveals too much semantic information relating to the original representative image. We see that the rest of the MarS are visually unintelligible. Upon discarding $\mathbf{u}_0$ (see security insights in Section 3.2) and $\mathbf{u}_{19}$, Figure 6 shows reconstruction using MarS $\{\mathbf{u}_i\}_{i=1}^{18}$. The

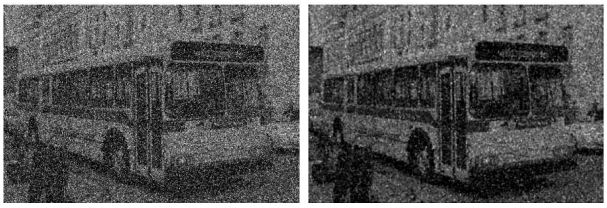

Figure 6. (a) Reconstruction using 18 Markov shares; (b) 3×3 median filtered version of (a)

missing 2 MarS $\{\mathbf{u}_0, \mathbf{u}_{19}\}$, and the additive Gaussian noise induces salt-and-pepper noise in the reconstructed image of Figure 6(a). Upon applying the 3×3 median filter, the reconstructed image is cleaned up, and reveals enough semantic information. From here-on, we shall apply this median filter to all authorized and unauthorized reconstructions.

Next we present the reconstructed image when each MarS is limited to having no more than 8 bits per pixel including a sign bit, and 2 floating point digits. We see that the reconstructed image in Figure 7 is no worse than that in Figure 6. In addition, there is a 1-bit saving per pixel

Figure 7. Reconstruction using 18 Markov shares and limiting number of bits per pixel

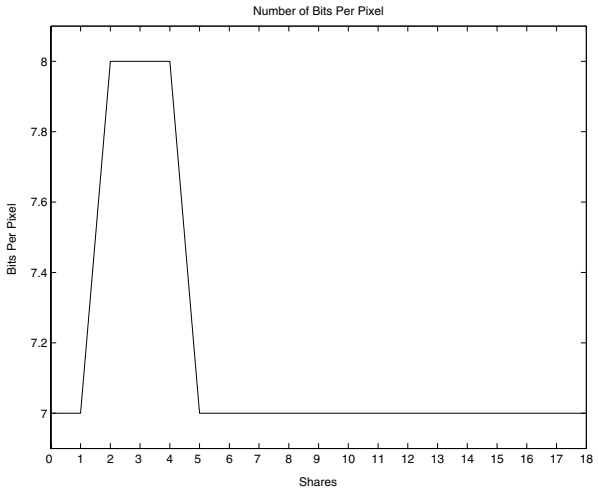

Figure 8. Bits per pixel of the bit-limited Markov shares

for most MarS as seen in Figure 8. Although this compression is not spectacular, we can increase our bit savings by lowering the threshold. However using the current parameters, we will soon see that robustness to errors from non-synchronized images, as well as denial of service attacks can be achieved.

We now relax our assumption that all the images acquired by the 20 sensor nodes are identical. In our simulations, we allowed each image acquired by individual sensor nodes to be a random rotation up to 10 degrees. We then let each sensor node create their own MarS without registering their image with any of the other sensors' images. Figure 9 shows that reconstruction without registration is feasible. In the next subsection, we show simulations pertaining to attacks. This will further demonstrate that our signal processing approach is capable of dealing with both unintentional

Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)
0-7695-2529-6/06 $20.00 © 2006 IEEE

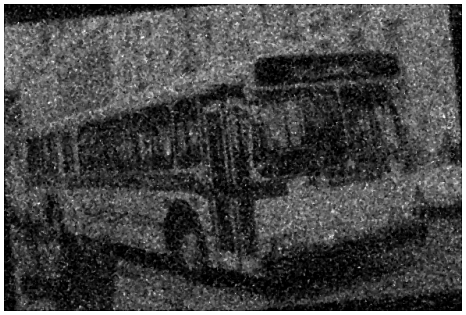

Figure 9. Reconstruction using unregistered Markov shares

Figure 11. Unauthorized reconstruction using first 5 Markov shares $\{\mathbf{u}_i\}_{i=1}^{5}$

and intentional errors.

4.1 Attacks

4.1.1 Single attack

First, we examine single MarS attacks, in which the attacker with a single MarS tries to recover the representative image using all necessary parameters. An attacker with MarS $\mathbf{u}_1$ can recover the representative image via the technique presented in Section 3.2. We stated that some light noise (0-mean, 0.05-variance additive Gaussian) was added to the first few MarS to hinder the attack. The best possible unauthorized approximation is shown in Figure 10. Here we

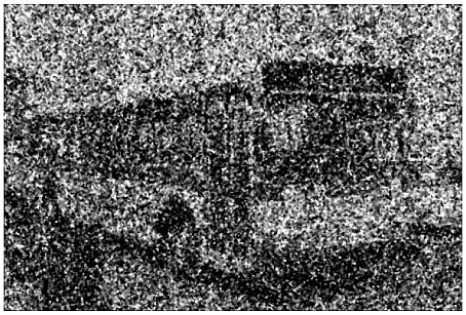

Figure 10. Unauthorized approximation using $\mathbf{u}_1$

have tried the Wiener and 3×3 median filters, with the latter giving better visual results.

4.1.2 Multiple attacks

Next we examine multiple MarS attacks, in which an attacker has approximately 5 MarS. The attacker's best attack is of course to obtain the first 5 MarS. Figure 11 shows the attacker's best multiple MarS attack using 5 MarS. Figure 12 shows attacks using 5 to 6 MarS. From our simu-

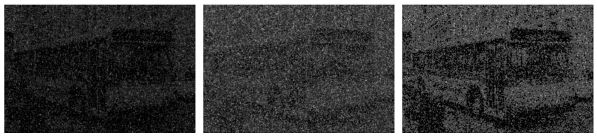

Figure 12. Unauthorized reconstructions using (a) $\mathbf{u}_9$ to $\mathbf{u}_{14}$; (b) $\mathbf{u}_{14}$ to $\mathbf{u}_{19}$; (c) $\mathbf{u}_2$, $\mathbf{u}_5$, $\mathbf{u}_{10}$, $\mathbf{u}_{15}$, $\mathbf{u}_{19}$

lation results, the attacker can gain access to crude semantic information when the number of MarS intercepted is about 5. For best attack results, the first few MarS need to be acquired. For example, in Figure 12(b), the last 5 MarS are used, and hence the quality of the attack suffers.

4.1.3 Denial of service attack

Suppose that an attacker is able to place an object to cover a region of interest of a single sensor node, hence corrupting this node's MarS. Will the other MarS be able to correct this corrupted region of interest? When is detection of such an attack possible? In Figure 13(a) a grizzly bear is placed to maliciously cover the front of the bus. First we show that if *all* MarS are transmitted, including the insecure first and last ones, then not only will correction be possible upon reconstruction, but visual detection is also possible. In Figure 13(b) it is possible to not only see the front of the bus, but also see the corrupting source (i.e. the semi-transparent grizzly bear). Hence for an application requiring both correction and detection, but not privacy/confidentiality, Markov shares can be used.

We now examine using only 18 MarS under the same parameters as all previous simulations. We see from Figure 14 that in each case the front of the bus is revealed,

Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)
0-7695-2529-6/06 $20.00 © 2006 IEEE

Figure 13. (a) Corrupted region of interest (grizzly bear © Photohome.com) of a single sensor node (b) detection of corruption using all Markov shares

Figure 14. Reconstruction given corrupted MarS (i.e. grizzly bear in front of MarS): (a) $\mathbf{u}_1$; (b) $\mathbf{u}_9$; (c) $\mathbf{u}_{18}$

and the corrupting source is removed. However, due to the noisy reconstruction, the semi-transparent grizzly bear is no longer visible to flag detection. Hence under our privacy/confidentiality application, only visual correction is possible, but not visual detection.

5 Conclusions and future work

We have shown that using only signal processing and control, a distributed obfuscation scheme for VSNs can be derived using Markov shares. Furthermore, using Markov shares offer robustness to both unintentional and intentional errors, such as non-registration, and tampering attacks respectively for example.

The drawback of our specific example is that the first few Markov shares are more prone to revealing semantic information than the other shares. In addition, reconstruction is not perfect when counter-measures are taken to hinder possible attacks against the aforesaid Markov shares. We envision that a marriage of coding with our signal processing approach would result in better quality reconstruction, as well as higher protection against the first few Markov shares, while offering the robustness benefits seen in this paper.

This paper also does not address security completely. For example, we focused only on our defined capture attack, and mentioned one heuristic attack. Future work will consider a more theoretical approach to addressing security, which will result in a framework suitable for analyzing security schemes based on Markov shares. In addition, the fuzzy definition of whether the semantics of an image are revealed or not will be addressed formally in future works.

Finally we remind the reader that changing the system and cost function will result in a different algorithm, one whose security would need to be studied. Hence there is potential for deriving a better system and cost metrics to improve both security as well as reconstruction fidelity.

References

[1] I. Agi and L. Gong. An empirical study of secure MPEG video transmissions. In *Proc. Internet Society Symposium on Network and Distributed System Security*, pages 137–144, San Diego, California, February 1996.

[2] G. Ateniese, C. Blundo, A. D. Santis, and D. R. Stinson. Visual cryptography for general access structures. *Information and Computation*, 129(2):86 – 106, September 1996.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 2 edition, 2001.

[4] L. Buttyàn and J.-P. Hubaux. Report on a working session on security in wireless ad hoc networks. *ACM Mobile Computing and Communications Review*, 6(4):1–17, November 2002.

[5] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *Proc. IEEE INFOCOM*, March 2004.

[6] M. Gerla and K. Xu. Multimedia streaming in large-scale sensor networks with mobile swarms. *ACM SIGMOD Record*, 32(32):72–76, December 2003.

[7] R. Ito, H. Kuwakado, and H. Tanaka. Image size invariant visual cryptography. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, E82-A(10):2172 – 2177, October 1999.

[8] C. C. Lin and W. H. Tsai. Secret image sharing with capability of share data reduction. *Optical Engineering., Bellingham (USA)*, 42(8):2340 – 2345, August 2003.

[9] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN Progress Report 42-44, Jet Propulsion Lab, 1978. pages 114 - 116.

[10] M. Naor and A. Shamir. Visual cryptography. In *Advances in Cryptology - EUROCRYPT '94. Workshop on the Theory and Application of Cryptographic Techniques. Proceedings*, pages 1 – 12, 1995.

[11] S. Pradhan and K. Ramchandran. Distributed source coding using syndromes (discus): Design and construction. *IEEE Trans. Information Theory*, 49:626–643, March 2003.

[12] L. Qiao and K. Nahrstedt. Comparison of MPEG encryption algorithms. *Computers and Graphics*, 22(4):437–448, 1998.

[13] A. Shamir. How to share a secret. In *Communications of the ACM 22*, number 11, pages 612–613, November 1979.

[14] V. Stankovic, A. Liveris, Z. Xiong, and C. Georghiades. Design of slepian-wolf codes by channel code partitioning. In *Proc. IEEE Data Compression Conference*, pages 302–311, March 2004.

[15] D. R. Stinson. *Cryptography: Theory and Practice*. Chapman and Hall, 1 edition, 1995.

Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)
0-7695-2529-6/06 $20.00 © 2006 IEEE