Video Surveillance: A Distributed Approach to Protect Privacy

Martin Schaffer and Peter Schartner

University of Klagenfurt, Austria
Computer Science · System Security
{m.schaffer, p.schartner}@syssec.at

Abstract. The topmost concern of users who are kept under surveillance by a CCTV-System is the loss of their privacy. To gain a high acceptance by the monitored users, we have to assure, that the recorded video-material is only available to a subset of authorized users under exactly previously defined circumstances. In this paper we propose a CCTV video surveillance system providing privacy in a distributed way using threshold multi-party computation. Due to the flexibility of the access structure, we can handle the problem of loosing private-key-shares that are necessary for reconstructing video-material as well as adding new users to the system. If a pre-defined threshold is reached, a shared update of the master secret and the according re-encryption of previously stored ciphertext without revealing the plaintext is provided.

1 Introduction

The major concern of users monitored by a CCTV video surveillance system is the loss of their privacy. It is obvious, that encrypting the recorded material raises the acceptance by the monitored users. But what about unauthorized decryption? In this paper we propose several mechanisms concerning the setup, the recording and the retrieval of videos, and the key management during all these phases. Some of the mechanisms involve multi-party computation (MPC, see [18,6,8]), so that we can enforce dual control. A trusted third party (TTP) may also be used to enforce dual control. But if this TTP is compromised, a single unauthorized person may be able to decrypt the whole video-material. The main requirements for our system include:

- Privacy-protection of the monitored users.
- Shared generation and update of keys and key components.
- Tree-based access structure to provide a mechanism for substitution.
- Dual control (4-eyes principle) within the video retrieval process.
- Minimal access of authorized people to monitored information.

Several papers about video surveillance exist, most of which focus on the ability to detect and identify moving targets. Only a few discuss the privacy protection of recorded material. The authors in [9] for example describe a cooperative, multi-sensor video surveillance system that provides continuous coverage

J. Dittmann, S. Katzenbeisser, and A. Uhl (Eds.): CMS 2005, LNCS 3677, pp. 140–149, 2005.

over battlefield areas. In [10] the emphasis lies on face recognition – a solution to protect privacy by de-identifying facial images is given. A similar approach can be found in [2] concentrating on videos in general. The most general solution to cover privacy in coherence to monitoring targets seems to be presented in [16]. However the system in [16] uses a privacy preserving video console providing access control lists. Once the console has been compromised videos may be decrypted without any restrictions.

In our paper we focus on video surveillance where real-time reactions are *not* necessary like in private organisations where staff has to be monitored. In case of criminal behaviour recorded video material can be decrypted if sufficiently enough instances agree – this e.g. is not provided in [10]. Our approach can certainly be combined with the general solution proposed in [16] but also used for other applications such as key escrow as proposed in [15].

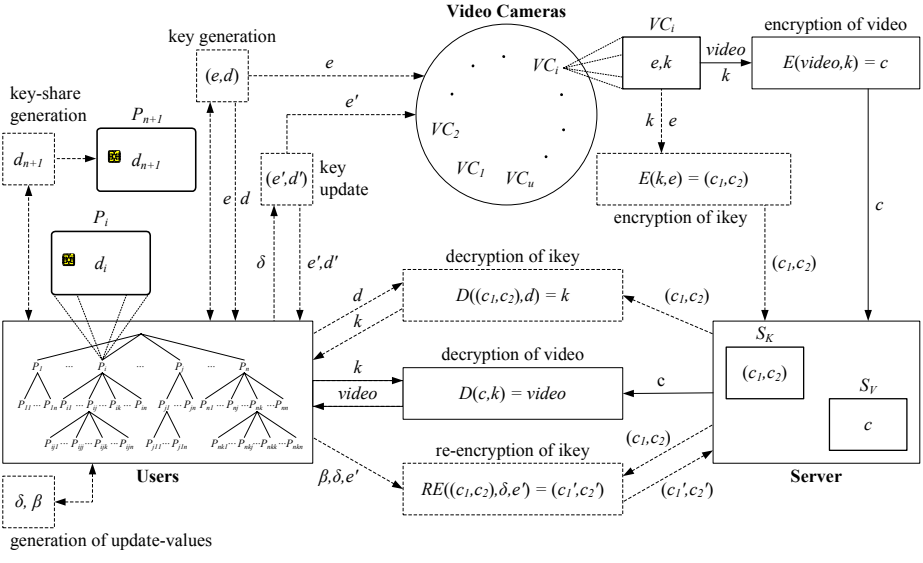

Fig. 1. System Architecture

Figure 1 shows the system architecture of the proposed system including all components and interactions within the setup phase, recording phase, retrieval phase and key management. Computations within dotted boxes represent MPCs whereas undotted boxes are performed by a single instance. Labelled arrows show which instance(s) deliver(s) or receive(s) which value(s).

The proposed system employs the following hardware-components and users:

Video Cameras. According to the monitored processes, video cameras record either single pictures or videos. Since we want to guarantee privacy of the monitored users, we have to encrypt the video material. After encrypting the video, it is sent to the video server S_V, whereas the key used for this encryption is (encrypted and) sent to the key server S_K.

Video Server S_V. Here the received encrypted video material is stored in a suitable database so that it can be easily retrieved, if required.

Key Server S_K. Keys that are used for encrypting videos are chosen interval-wise (so we provide minimal access to videos). Therefore, we call them interval-keys (ikeys) and store them encrypted at S_K.

Users. In this paper, the term *user* always means an instance retrieving videos, not a person monitored by the system. Within strict regulations (e.g. dual control), these users are authorized to decrypt the stored videos. Due to the fact that enterprises are often hierarchically structured, we have to provide easy deputy-mechanisms. If a user is not available, he may be simply replaced by a qualified set of users of the next lower level in the hierarchy.

Smartcards. In order to enforce the cooperation of several users in the decryption process, the corresponding private key d is shared among a group of users. Hence, each user holds a share of the private key, which is stored in a pin-protected smartcard. Note that the private key d is never available in the system or present during intermediate results of the decryption process.

The proposed system consists of the following procedures:

Setup Phase. To initialize the system, the users perform a MPC which provides each user with a random share of private key d in a fair way. Additionally, the users generate shares of the corresponding public key. These shares are sent to the video cameras which reconstruct the public key e.

Recording Phase. Due to performance reasons, the recorded video-material is encrypted by use of a hybrid cryptosystem. Hence, each video camera has to hold the public key e. The ikey is then encrypted by use of an asymmetric scheme (employing the public key e) and is finally sent to S_K, whereas the symmetrically encrypted video is sent to S_V.

Retrieval Phase. Within the retrieval phase, the authorized users perform a MPC which provides them with the ikey of the specific interval, without direct usage of the private key d (which corresponds to the public key e).

Key Management. In order to take part in the system, a new user has to retrieve a share of the private key d. To achieve this, the users already enrolled in the system perform a MPC which finally provides the new user with his share. Since the decryption process involves threshold cryptography, some smartcards (and the shares stored there) may be lost, without any danger for the privacy of the stored video material. Additionally, we employ a mechanism which regularly updates the remaining shares (without changing the shared private key) and hence makes the shares on the lost (or stolen) smartcards useless. Finally, we propose a mechanism to perform a shared update of d and the corresponding public key e. It is obvious that in this case, all encrypted ikeys have to be re-encrypted, whereas the encrypted video material remains unchanged since the ikeys have not been compromised.

In the remainder of the paper we will give a more formal description of the processes briefly discussed by now. Note that within the proposed mechanisms we will only care about passive adversaries (see [6,8]) from inside the system and

we will assume that there exist pair-wise protected links between the individual parties participating in the surveillance process.

2 Fundamentals

Every computation in the following sections – except symmetric algorithms – is reduced modulo p (within bases) or modulo q (within exponents, sharing polynomials and interpolation formulas). The direct successors of the root of the tree-based access structure are called first-level-users and united in the set $\mathcal{U}$. To reduce complexity, we will only consider one video camera called VC.

2.1 Shamir's Secret Sharing

To share a secret $s \in \mathbb{Z}_q^*$ among n users resulting in the shares $s_1, \ldots, s_n$ (short: $s \mapsto (s_1, \ldots, s_n)$) we use the following randomly chosen t-degree polynomial according to [17]:

$$s_i = g(i), \quad g(x) = s + \sum_{j=1}^{t} r_j \cdot x^j, \quad r_j \in_R \mathbb{Z}_q^* \tag{1}$$

In order to reconstruct the secret s (short: $(s_1, \ldots, s_n) \mapsto s$) we need at least $t+1$ shares, because there are $t+1$ unknown values in a t-degree polynomial. For efficiency reasons we use the interpolation formula of Lagrange (see e.g. [12]):

$$s = g(0), \quad g(x) = \sum_{i=1}^{n} s_i \cdot \lambda_{x,i}^s, \quad \lambda_{x,i}^s = \prod_{\substack{j=1 \\ j \neq i}}^{n} (x-j) \cdot (i-j)^{-1} \tag{2}$$

Several computations of the upcoming sections use the following transformation:

$$z = y^s \overset{(2)}{=} y^{\sum_{i=1}^{n} s_i \cdot \lambda_{0,i}^s} = \prod_{i=1}^{n} y^{s_i \cdot \lambda_{0,i}^s} \tag{3}$$

2.2 Symmetric Cryptosystem

Recording videos causes a lot of data. Hence, we apply a symmetric algorithm (e.g. AES, see [1]) to encrypt the video-material. We simply define the encryption function $E_S(m,k) = c$ and decryption function $D_S(c,k) = m$.

2.3 ElGamal Cryptosystem

We suppose that the reader is familiar with the basic ElGamal cryptosystem [4]. Assuming the key generation has already taken place resulting in the public key e and the private key d, the encryption E and decryption D can be performed as follows (with g a generator of $\mathbb{Z}_q^*$):

$$E(m,e) = (g^\alpha, m \cdot e^\alpha) = (c_1, c_2), \quad e = g^d, \quad \alpha \in_R \mathbb{Z}_q^* \tag{4}$$

$$D((c_1,c_2),d) = c_2 \cdot \left(c_1^d\right)^{-1} = m \tag{5}$$

3 ElGamal Threshold Decryption and Re-encryption

A public key cryptosystem can be shared in several ways. The plaintext, the public key, the ciphertext as well as the private key can be used in a distributed way. For a video surveillance system sharing the encryption process does not make sense. However, sharing the decryption process enables us to realize dual-control. To increase the security it is very useful to share the ciphertext as well. For lack of space we decided not to describe this variation. Instead we focus on how to share the decryption process emphasizing on selected aspects of the corresponding management of key-shares. Due to its simplicity we use ElGamal threshold decryption firstly proposed in [3]. The basic ElGamal decryption can be divided into two parts so that its computation only uses shares of private key d. Therefore d has to be shared using a t-degree polynomial: $d \mapsto (d_1, \ldots, d_n)$. The ElGamal decryption function can be modified replacing d with its Lagrange-representation over the shares:

$$D((c_1, c_2), d) = c_2 \cdot \left(c_1^d\right)^{-1} \overset{(3)}{=} c_2 \cdot \left(\prod_{i=1}^{n} c_{1i}^{\lambda_{0,i}^d}\right)^{-1} \overset{(5)}{=} m, \quad c_{1i} = c_1^{d_i} \tag{6}$$

Now we can divide this computation into the following two sub-functions:

Decryption Step 1. This step has to be done by at least $t+1$ shareowners.

$$D_1(c_1, d_i) = c_1^{d_i} \overset{(6)}{=} c_{1i}$$

Decryption Step 2. To compute m at least $t+1$ outputs of D_1 are required.

$$D_2((c_{11}, \ldots, c_{1n}), c_2) = c_2 \cdot \left(\prod_{i=1}^{n} c_{1i}^{\lambda_{0,i}^d}\right)^{-1} \overset{(6)}{=} m$$

If the private key d has been compromised we have to provide an update of d and a re-encryption of the corresponding ciphertext without revealing plaintext. In [19] an approach based on distributed blinding is given. There, a ciphertext is first blinded by a randomly chosen and encrypted value. After having decrypted the blinded ciphertext in a particular way the resulting blinded plaintext is encrypted with the new public key and finally unblinded. The advantage of this approach is that the instances that blind the ciphertext do not know anything about the private key. This is useful for transferring a ciphertext from one instance to another one (with different keys). In our case we need a mechanism that provides an update of the private key and the corresponding ciphertext. In our scenario the solution in [19] would require a distributed blinding, a distributed decryption and a distributed unblinding. As a consequence we propose a different variation based on the distance δ between the old private key d and the new private key d'. The advantage of our re-encryption is that we only modify the old ciphertext and do not perform decryptions and encryptions respectively.

Theorem 1. *Assume (c_1, c_2) is a ciphertext performed over m and e. Then a ciphertext based on $e' = e \cdot g^{\delta}$ and decryptable by $d' = d + \delta$ can be computed by doing the following random transformation of (c_1, c_2) without intermediately revealing the corresponding plaintext m:*

$$RE((c_1, c_2), \delta, e') = (c_1 \cdot g^{\beta}, c_2 \cdot c_1^{\delta} \cdot e'^{\beta}) = (c_1', c_2'), \quad \beta \in_R \mathbb{Z}_q^* \tag{7}$$

$$d' = d + \delta, \quad e' = e \cdot g^{\delta} \tag{8}$$

Proof. Let (c_1', c_2') be a transformed ciphertext according to (7). Then the basic ElGamal decryption with new private key d' results in m because:

$$\begin{aligned} D((c_1', c_2'), d') &\overset{(5)}{=} c_2' \cdot (c_1'^{d'})^{-1} \overset{(7)}{=} c_2 \cdot c_1^{\delta} \cdot e'^{\beta} \cdot ((c_1 \cdot g^{\beta})^{d'})^{-1} \\ &\overset{(8)}{=} c_2 \cdot c_1^{\delta} \cdot (e \cdot g^{\delta})^{\beta} \cdot ((c_1 \cdot g^{\beta})^{d+\delta})^{-1} \\ &\overset{(4)}{=} m \cdot e^{\alpha} \cdot g^{\alpha \cdot \delta} \cdot (g^d \cdot g^{\delta})^{\beta} \cdot ((g^{\alpha} \cdot g^{\beta})^{d+\delta})^{-1} \\ &\overset{(4)}{=} m \cdot g^{\alpha(d+\delta)} \cdot g^{\beta(d+\delta)} \cdot (g^{(\alpha+\beta)(d+\delta)})^{-1} = m \end{aligned}$$

□

The re-encryption process in (7) can also be divided into two sub-functions so that it can be performed in a distributed way (assume: δ and β are shared):

Re-encryption Step 1. The first step is done locally by every user P_i.

$$RE_1(c_1, \delta_i, e', \beta_i) = (g^{\beta_i}, c_1^{\delta_i}, e'^{\beta_i}) = (\tilde{c}_{1i}, c_{1i}, e_i')$$

Re-encryption Step 2. The second step uses all outputs of RE_1 and the old ciphertext.

$$RE_2(c_1, (\tilde{c}_{11}, \ldots, \tilde{c}_{1n}), (c_{11}, \ldots, c_{1n}), (e_1', \ldots, e_n'), c_2) = (c_1', c_2')$$

$$c_1' = c_1 \cdot \prod_{i=1}^{n} \tilde{c}_{1i}^{\lambda_{0,i}^{\beta}}, \quad c_2' = c_2 \cdot \left(\prod_{i=1}^{n} c_{1i}^{\lambda_{0,i}^{\delta}} \right) \cdot \prod_{i=1}^{n} e_i'^{\lambda_{0,i}^{\beta}}$$

If there is no need to mask the correspondence between old and new ciphertext, the modifications of the original randomness α by use of β can be removed.

4 Video Surveillance

4.1 Setup Phase

During the initialization of the system, a key-pair (e, d) for the ElGamal cryptosystem has to be generated in a shared way. To achieve this, all users cooperatively generate shares of the private key d without reconstructing it. Then they compute shares of e without any interaction and send them to the video camera which interpolates e. The distributed key generation proposed in [11] is very useful to generate a private key without reconstructing it. A more secure version is proposed in [5]. However, we need a fair tree-structured generation of the private key. Based on this fact we modify the original protocol in order to be able to build such a tree. A detailed description of a tree-shared generation of secret values can be found in [14] – we refer to it for lack of space.

4.2 Recording Phase

Within this phase VC uses local hybrid encryption. First of all VC generates an interval-key k at random and encrypts the interval-video using symmetric encryption described in section 2.2: $E_S(video, k) = c$. For encryption of k the camera uses asymmetric encryption described in section 2.3 with public key e: $E(k, e) = (c_1, c_2)$. Within each interval the camera sends the encrypted video to S_V and its corresponding encrypted ikey to S_K. Both server store the ciphertext in a particular database.

4.3 Retrieval Phase

The retrieval of a particular video can be done in two steps:

Decryption of ikey. S_K has to send (c_1, c_2) to every user in $\mathcal{U}$ who agrees to reconstruct the video. Then each user P_i performs $D_1(c_1, d_i) = c_{1i}$ and broadcasts the result within $\mathcal{U}$. Finally every user P_i decrypts ikey k by computing $D_2((c_{11}, \ldots, c_{1n}), c_2) = k$.

Decryption of Video. S_V has to send the encrypted video c (corresponding to k) to every user P_i who decrypts it by performing $D_S(c, k) = video$.

5 Managing Private-Key-Shares

Generally, an access structure has to be very flexible within an organisation. The more users exist the sooner it might occur that a user is leaving or joining the system.

5.1 Registration of a New User

When registering a new user P_{n+1} we have to distinguish users of the first level who do not have a predecessor and users of lower levels who always have predecessors.

New First-Level-User. Every existing first-level-user P_i shares his share $d_i \mapsto (d_{i1}, \ldots, d_{in+1})$ among $\mathcal{U}' = \mathcal{U} \cup \{P_{n+1}\}$. Then every user P_j in $\mathcal{U}'$ interpolates the received shares $(d_{1j}, \ldots, d_{nj}) \mapsto d_j$. Due to the fact, that every share changes, an update of successor-shares has to be performed.

Others. Every user P_i of a lower level always has a predecessor P who is responsible for registering his new successor P_{n+1}. If P does not know the shares of his existing successors they have to send him their shares. Owning at least $t + 1$ shares of his successor enables P to generate a share $d_{n+1} = \sum_{i=1}^{n} d_i \cdot \lambda_{n+1,i}^{d}$ for P_{n+1} without provoking a recursive update of successor-shares. After importing d_{n+1} to P_{n+1}'s smartcard P removes $d_1, \ldots, d_n$ form his smartcard.

Generation of new shares can be done in several ways. An important fact is to keep side effects minimal which we cannot guarantee with the solution described above when registering a first-level-user. For more efficient but also some more complex variations we refer to our technical report [13].

5.2 Loss of Smartcards

If a user collects at least $t + 1$ previously lost smartcards he might be able to compromise the private key d. Regularly updates of shares without changing d make the collector's shares unusable. Such updates can be very time-consuming because all users of the access structure have to participate in the update process at the same time (except if centralized updates are used). If a user looses his smartcard his share can be reconstructed using the computations in section 5.1. We always have to consider the worst case which is that another user of the access structure finds the smartcard. Then the threshold is decreased which we want to avoid. Due to this fact we propose to run an update-protocol first and then generate a new share for the user who lost his smartcard.

5.3 Proactive Behaviour

Collecting lost smartcards can be used to decrease the threshold. So we have to update the private-key-shares without changing the private key (as proposed in [7]). This should be done in case of loosing a smartcard but can also be performed proactively regularly. Using short intervals can be very time-consuming if updates are done in a distributed way because users have to be online at the same time. In this case the update could be initiated by a central trusted authority. A big advantage of this variation is that updates could be run in batch-mode.

What happens if threshold t is vulnerable within one interval? In this case we propose to update the private key in a shared way in sufficient time which forces a re-encryption of ciphertext that corresponds to the compromised private key (see section 6). Until the re-encryption process has been finished S_K has to be protected against availability-compromising attacks. To handle this problem we propose to share the ciphertext-pairs (c_1, c_2) among several server. This would lead to several modifications of the basic system which we do not describe here.

5.4 De-registration of Users

If a user leaves the organisation his smartcard (holding the share) should be securely destroyed. If a new user takes over his tasks the protocol described in section 5.1 has to be run.

6 Update of Private Key and Corresponding Ciphertext

First of all (e, d) has to be updated by all cameras and all shareowners of d. Before destroying the update-values a re-encryption of every ciphertext (c_1, c_2) generated using e has to be done.

Shared Generation of Update-Values. All the users in $\mathcal{U}$ run the tree-based key generation mentioned in section 4.1 to get shares $\delta_1, \ldots, \delta_n$ of a private-key-update δ and shares $\beta_1, \ldots, \beta_n$ of randomness-update β.

Update of Private-Key-Shares. Every user P_i computes $d'_i = d_i + \delta_i$ which is a share of private key $d' = d + \delta$.

Shared Update of Public Key. All users perform the updated public key $e' = e \cdot \prod_{i=1}^{n} g^{\delta_i \cdot \lambda^{\delta}_{0,i}}$ in a distributed way and send e' to VC.

Re-encryption of Encrypted ikeys. S_K has to send the old ciphertext-part c_1 to every user P_i participating in the re-encryption processes. Then each P_i has to perform $RE_1(c_1, \delta_i, e', \beta_i) = (\tilde{c_{1i}}, c_{1i}, e'_i)$. Finally, each output of RE_1 has to be sent to S_K which then replaces the old ciphertext (c_1, c_2) by the output of $RE_2(c_1, (\tilde{c}_{11}, \ldots, \tilde{c}_{1n}), (c_{11}, \ldots, c_{1n}), (e'_1, \ldots, e'_n), c_2) = (c'_1, c'_2)$.

7 Security Analysis

We now briefly analyse the power of each instance of the system to retrieve any secret information. However, we do not consider the tree-structure – the analysis can be interpreted recursively.

As long as video cameras are not able to solve the discrete logarithm problem and do not compromise at least $t + 1$ first-level-users, they are not able to get any information about the private key d, update-values δ and β or any shares of the users. To decrypt video-material S_V needs the corresponding ikey. But to get access to it he has to compromise at least $t + 1$ first-level-users and S_K. A first-level-user needs at least t other shares to reconstruct d or update-values δ and β. Moreover, he has to compromise S_K and S_V to be able to decrypt videos. Up to t smartcards of first-level-users can be stolen and compromised without revealing any information about d. Regularly updates of shares increase the security of the private key. Moreover, the smartcards are secured by a Personal Identification Number. To preserve resistance against active malicious behaviour (e.g. sending wrong intermediate results), extensions according to secure multi-party computation with active adversaries are required (see [6,8]).

8 Conclusion and Future Research

Considering the requirements stated in section 1, it can be seen that all of them have been realized.

Privacy-protection of the monitored users is provided by encryption of video-material and interval-keys. 4-eyes principle (dual control) is provided by a tree-based access structure. Minimal access of authorized people to monitored information is guaranteed by scaling monitored intervals to a minimum so that many ikeys are generated. Keys and key components are generated tree-based in a fair distributed way according to [14]. Update of keys and key components is realized by tree-based update-value generation and threshold re-encryption. Tree-based secret sharing provides the possibility to replace any user by his successors.

The discussed distributed version of ElGamal is well known since [3] and only one-out-of many. Discussing how public key cryptosystems can be distributed can lead to many more applications than access structures to monitored information.

When sharing functions the management of key-shares appears to be much more difficult than the "normal" key management. So our future research work will emphasize on managing keys in distributed public-key cryptosystems keeping the number of local shares minimal not limiting to the ElGamal cryptosystem.

References

1. Advanced Encryption Standard (AES). FIPS-Pub 197, NIST, 2001.
2. M. Boyle, C. Edwards, S. Greenberg. The Effects on Filtered Video on Awareness and Privacy. CSCW'00: Proceed. of the 2000 ACM conference on Computer supported cooperative work, Philadephia, PA, USA, 2000, pages 1–10.
3. Y. Desmedt, Y. Frankel. Threshold Cryptosystems. Adv. in Crypt.: Proceed. of CRYPTO'89, Springer-Verlag, pp. 307–315, 1990.
4. T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. Adv. in Crypt.: CRYPTO'84, Springer-Verlag, pp. 10–18, 1985.
5. R. Gennaro et al. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. Proceed. of EUROCRYPT'99, Springer LNCS 1592, pp. 295-310.
6. O. Goldreich et al. How to play any mental game – a completeness theorem for protocols with honest majority. Proc. 19th ACM STOC, p. 218–229, 1987.
7. A. Herzberg et al. Proactive secret sharing or: how to cope with perpetual leakage. In Adv. in Crypt.: CRYPTO'95, vol. 963 of LNCS, Springer-Verlag, pp. 339–352.
8. M. Hirt. Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting. Ph.D. thesis, ETH Zurich, 2001 Reprint as vol. 3 of ETH Series in Information Security and Cryptography, Hartung-Gorre Verlag, Konstanz, 2001.
9. R.T. Collins et al. A System for Video Surveillance and Monitoring. Proceedings of the American Nuclear Society (ANS) Eighth International Topical Meeting on Robotics and Remote Systems, 1999.
10. E. Newton, L. Sweeney, B. Malin. Preserving Privacy by De-identifying Facial Images. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 2, pp. 232–243, February 2005.
11. T. Pedersen. A threshold cryptosystem without a trusted party. Adv. in Crypt.: EUROCRYPT'91, vol. 547 of LNCS, Springer-Verlag, pp. 522–526, 1991.
12. J. Pieprzyka, T. Hardjono, J. Seberry. Fundamentals of Computer Security. Springer-Verlag, 2003.
13. M. Schaffer. Managing Key-Shares in Distributed Public-Key Cryptosystems. Technical Report TR-syssec-05-04, University of Klagenfurt, Austria, August 2005.
14. M. Schaffer. Tree-shared Generation of a Secret Value. Technical Report TR-syssec-05-01, University of Klagenfurt, Austria, June 2005.
15. M. Schaffer, P. Schartner. Hierarchical Key Escrow with Passive Adversaries. Technical Report TR-syssec-05-02, University of Klagenfurt, Austria, June 2005.
16. A. Senior et al. Blinkering Surveillance: Enabling Video Privacy through Computer Vision. IBM Research Report RC22886 (W0308-109), August 28, 2003.
17. A. Shamir. How to share a secret. Comm. of the ACM, vol. 11, pp. 612–613, 1979.
18. A.C. Yao. Protocols for secure computation. In Proceed. of the 23rd IEEE Symposium on Foundations of Computer Security, 1982.
19. L. Zhou et al. Distributed Blinding for ElGamal Re-encryption. Proceed. 25th IEEE Int. Conf. on Distributed Computing Systems. Ohio, June 2005, p. 815–824.