if $\{(\mathbf{b}_{t,x,y} \in$ dynamic region)

and $(\mathbf{b}_{t,x-v_x,y-v_y} \in$ projected region)$\}$

$\mathbf{b}_{t,x,y}$ is a false motion block

else

$\mathbf{b}_{t,x,y}$ is a true motion block (3)

If the number of true motion blocks is larger than the number of false motion blocks in a projected region, the region becomes a projected dynamic region. The uncovered region has a smaller number of true motion blocks than of false motion blocks in the projected region.

Finally, moving objects consist of dynamic regions and projected dynamic regions.

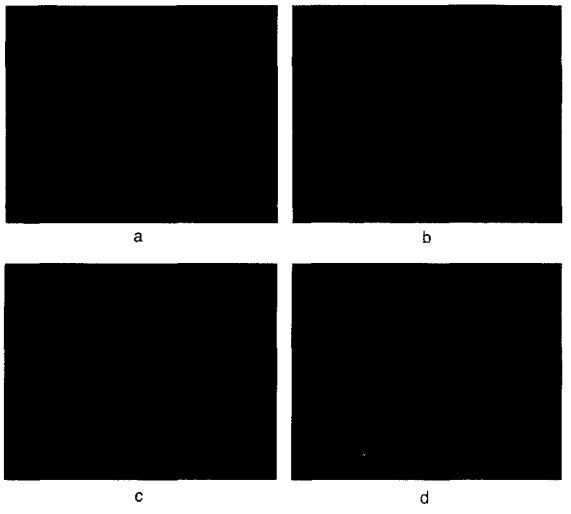


Fig. 2 *Blocks with non-zero motion vectors in 'mother and daughter' sequence*

a 3rd frame
b 81st frame
c 180th frame
d 213th frame

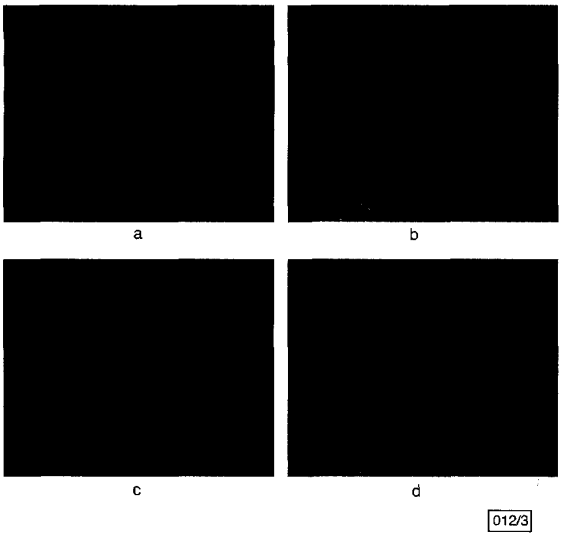


Fig. 3 *Segmented moving object of 'mother and daughter' sequence from proposed segmentation method*

a 3rd frame
b 81st frame
c 180th frame
d 213th frame

Experimental results: Experiments were performed on several compressed bit-streams. Figs. 2 and 3 show blocks with non-zero motion vectors and a segmented moving object, respectively, for the 'mother and daughter' QCIF sequence, which was compressed by an H.263 encoder with a frame rate of 10Hz and a target bitrate of 24kbit/s. The frame numbers in Figs. 2 and 3 are from the original sequence obtained at a frame rate of 30Hz. Although motion vectors themselves are not appropriate for moving object segmentation as shown in Fig. 2, the proposed segmentation method is able to detect and track moving objects, as shown in Fig. 3.

Discussion: We have developed a block-based moving object segmentation algorithm for compressed video. Since block-based video coders determine motion vectors based on the coding efficiency, motion vectors may give false motion information. However, the proposed algorithm uses the stochastic behaviour of spatially similar blocks to segment moving objects and the segmentation result is successful.

29 August 2000
Electronics Letters Online No: 20001279
DOI: 10.1049/el:20001279

Salkmann Ji and HyunWook Park (*Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, 373-1 Kusong-dong, Yusong-gu, Taejon 305-701, Korea*)

E-mail: hwpark@athena.kaist.ac.kr

References

1 CHANG, S.F., and MESSERSCHMITT, D.G.: 'Manipulation and compositing of MC-DCT compressed video', *IEEE Trans. Commun.*, 1995, **13**, (1), pp. 1–11
2 DOGAN, S., SADKA, A.H., and KONDOZ, A.M.: 'Efficient MPEG-4/H.263 video transcoder for interoperability of heterogeneous multimedia networks', *Electron. Lett.*, 1999, **35**, (11), pp. 863–864
3 JAIN, A.K.: 'Fundamentals of digital image processing' (Prentice-Hall, 1989)

Cryptanalysis of modified authenticated key agreement protocol

Wei-Chi Ku and Sheng-De Wang

Tseng addressed a weakness within and proposed a modification to the key agreement protocol presented by Seo and Sweeney. The authors show that Tseng's modified protocol is still vulnerable to two simple attacks and describe a new enhancement to the Seo-Sweeney protocol.

Introduction: By using a pre-shared password technique, Seo and Sweeney [1] proposed a simple key agreement protocol which was intended to act as a Diffie-Hellman scheme [2] with user authentication. In the Seo-Sweeney protocol, two parties who have shared a common password can establish a session key by exchanging two messages. The authors also claimed that key validation can be achieved by exchanging two more messages. Later, Tseng [3] addressed a weakness in the key validation steps of the Seo-Sweeney protocol. By replying to the message sent from the honest party, the adversary can fool the honest party into believing a wrong session key. Tseng modified the key validation steps of the Seo-Sweeney protocol and claimed that key validation can be achieved in the modified protocol. In this Letter, we will show that Tseng's modified protocol is still vulnerable to two simple attacks. Additionally, a new enhancement to the Seo-Sweeney protocol will be described.

Tseng's modified protocol: As in the original Diffie-Hellman scheme [2], the system possesses two public values n and g, where n is a large prime and g is a generator with order $n-1$ in $GF(n)$. Let Alice and Bob denote the two parties who have shared a common password P. The protocol has two phases, the key establishment phase and the key validation phase, and can be described as follows:

Key establishment phase:

(*e.1*) Alice and Bob each compute two integers Q and Q^{-1} mod (n

– 1) from P, where Q is computed in a predetermined way and is relatively prime to $n - 1$.

(*e.2*) Alice selects a random integer a and sends Bob

$$X_1 = g^{aQ} \bmod n$$

(*e.3*) Bob also selects a random integer b and sends Alice

$$Y_1 = g^{bQ} \bmod n$$

(*e.4*) Alice computes the session key Key_1 as follows:

$$Y = Y_1^{Q^{-1}} \bmod n = g^b \bmod n$$
$$Key_1 = Y^a \bmod n = g^{ab} \bmod n$$

(*e.5*) Bob computes the session key Key_2 as follows:

$$X = X_1^{Q^{-1}} \bmod n = g^a \bmod n$$
$$Key_2 = Y^b \bmod n = g^{ab} \bmod n$$

Key validation phase:

(*v.1*) Alice sends Y to Bob.

(*v.2*) Bob sends X to Alice.

(*v.3*) Alice and Bob check whether $X = g^a \bmod n$ and $Y = g^b \bmod n$ hold or not, respectively.

Backward replay without modification [4]: Upon seeing X_1 sent by Alice in step (*e.2*), the adversary (Eve) can masquerade as Bob to re-send it back to Alice in step (*e.3*) as Y_1. Consequently, Alice will compute

$$Y = Y_1^{Q^{-1}} \bmod n = X_1^{Q^{-1}} \bmod n = g^a \bmod n$$
$$Key_1 = Y^a \bmod n = g^{a^2} \bmod n$$

and send Y to Bob in step (*v.1*). Then, Eve can masquerade as Bob to re-send Y back to Alice in step (*v.2*) as X. Since $Y = g^a \bmod n$ holds, Alice will be fooled into believing the wrong session key Key_1. It should be noted that if step (*v.1*) and step (*v.2*) are exchanged, the protocol is still vulnerable to the replay attack, in which Eve masquerades as Bob to start another protocol run with Alice by using X_1. The message sent by Alice in the first key validation step of the new protocol run can be used by Eve in the second key validation step of the original protocol run. Again, Alice will be fooled into believing the wrong session key.

Modification attack: Upon seeing X_1 sent by Alice in step (*e.2*), Eve can replace it with any number $\in [1, n - 1]$, say X_1'. In step (*e.3*), Bob sends Y_1 to Alice, and then Alice sends the corresponding response Y to Bob in step (*v.1*). In step (*v.2*), Bob will send X', which equals $(X_1')^{Q^{-1}} \bmod n$, to Alice. Because $X' \neq g^a \bmod n$, Alice will not believe Key_1. However, since $Y = g^b \bmod n$ holds, Bob will believe the wrong session key Key_2', which equals $(X_1')^{Q^{-1}b} \bmod n$. Although Eve cannot compute Key_2', she can still fool Bob into believing this wrong session key. Note that if step (*v.1*) and step (*v.2*) are exchanged, the protocol is still vulnerable to the modification attack in the opposite direction, i.e. it is Alice rather than Bob who will be fooled into believing a wrong session key.

Enhanced key validation steps:

(*v.1*) Alice computes

$$Y_2 = (Key_1)^Q \bmod n = g^{abQ} \bmod n$$

and then sends Y_2 to Bob.

(*v.2*) Bob checks whether $Y_2^{Q^{-1}} \bmod n = Key_2$ holds or not. If it holds, Bob believes that he has obtained the correct X_1 and Alice has obtained the correct Y_1, i.e. Bob is convinced that Key_2 is validated, and then sends X to Alice.

(*v.3*) Alice checks whether $X = g^a \bmod n$ holds or not. If it holds, Alice believes that she has obtained the correct Y_1 and Bob has obtained the correct X_1, i.e. Alice is convinced that Key_1 is validated.

Discussions: The weakness of the Seo-Sweeney protocol is due to the same values of the two key validation messages. One problem within Tseng's modified protocol is that the values of the two key validation messages will be the same once $Y_1 = X_1$. Another problem within Tseng's modified protocol is that Bob cannot judge the correctness of X_1 from the received Y. In the enhanced key validation steps, the first key validation message is directly inherited from the Seo-Sweeney protocol while the second key validation message is adopted from Tseng's modified protocol. The use of asymmetric messages in the enhanced key validation steps is one of the methods of resisting the attack of backward replay without modification [4]. In addition, the first key validation message, Y_2, can alternatively be generated from $Y_2 = (Y_1)^a \bmod n$ and verified by checking whether $Y_2 = (X_1)^b \bmod n$. This alternative is useful if the protocol is implemented in hardware. As the generation (or verification) of Y_2 can be performed in parallel with the session key generation, the computation delay can be reduced.

4 September 2000
Electronics Letters Online No: 20001269
DOI: 10.1049/el:20001269

Wei-Chi Ku and Sheng-De Wang (*Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, Republic of China*)

E-mail: sdwang@hpc.ee.ntu.edu.tw

References

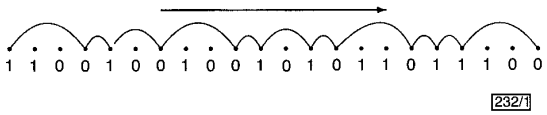
1 SEO, D.H., and SWEENEY, P.: 'Simple authenticated key agreement algorithm', *Electron. Lett.*, 1999, **35**, (13), pp. 1073–1074
2 DIFFIE, W., and HELLMAN, M.E.: 'New directions in cryptography', *IEEE Trans.*, 1976, **IT-22**, (6), pp. 644–654
3 TSENG, Y.M.: 'Weakness in simple authenticated key agreement protocol', *Electron. Lett.*, 2000, **36**, (1), pp. 48–49
4 GONG, L.: 'Variations on the themes of message freshness and replay'. Proc. IEEE Computer Security Foundations Workshop VI, June 1993, pp. 131–136

Embedding attacks on step[1..*D*] clock-controlled generators

W.G. Chambers and D. Gollmann

In a step[1..*D*] cryptographic generator a selector determines which bits from a primitive shift-register's output are sent to the final output, the maximum spacing being *D*. Two attacks are described, one through finding where embeddings are possible, valid for $D = 2$ and 3, and the other through counting embeddings.

Introduction: A clock-controlled cryptographic sequence generator produces as its final output $\{y_i\}_0^s$ the irregular decimation of a binary sequence $\{x_i\}$ produced by a pseudorandom binary generator A. (For the sake of definiteness we assume that A is a primitive linear feedback shift register (LFSR) of period N). The decimation is controlled by another pseudorandom generator S (the 'selector') which in effect gives rise to a strictly increasing series of integers $a[i]$ such that $y_i = x_{a[i]}$ for $i = 0, 1, 2, ..., s$. We say that the sequence $\{y_i\}_0^s$ can be embedded in $\{x_i\}$ at the location $a[0]$. If, moreover, we require that $a[i] - a[i-1] \leq D$ for some fixed integer D ($D > 1$) then we call the embedding a step[1..*D*] embedding. Fig. 1 shows how the sequence {101001100110} of length $s + 1 = 12$ can be embedded in a 'target' sequence of length $e + s + 1 = 22$. There are $e = 10$ points skipped ('skips') and $s + 1 = 12$ 'hits'. Note that the embedding starts and ends with hits.



232/1

Fig. 1 *Embedding of sequence $y = \{101001100110\}$ of length $s + 1 = 12$ in target sequence of length $e + s + 1 = 22$*

There are $e = 10$ points skipped ('skips') and $s + 1 = 12$ hits. The embedding starts and ends with hits. Note that there are several possible embeddings of y into the sequence shown

In an embedding attack we assume that we know a prefix of the final output $\{y_i\}_0^s$ for sufficiently large s, and we wish to find out