Some Notes on the Security of the Timed Efficient Stream Loss-Tolerant Authentication Scheme

Goce Jakimoski*

Department of Electrical and Computer Engineering
Stevens Institute of Technology, Burchard 212, Hoboken, NJ 07030, USA

Abstract. RFC4082 specifies the Timed Efficient Stream Loss-tolerant Authentication (TESLA) scheme as an Internet standard for stream authentication over lossy channels. In this paper, we show that the suggested assumptions about the security of the building blocks of TESLA are not sufficient. This can lead to implementations whose security relies on some obscure assumptions instead of the well-studied security properties of the underlying cryptographic primitives. Even worse, it can potentially lead to insecure implementations. We also provide sufficient security assumptions about the components of TESLA, and present a candidate implementation whose security is based on block ciphers resistant to related-key cryptanalysis.

Keywords: message authentication, multicast stream authentication, TESLA, cryptanalysis, block ciphers, related-key attacks.

1 Introduction

While most network applications are based on the client-server paradigm and make use of point-to-point packet delivery, many emerging applications are based on the group communications model. In particular, a packet delivery from one or more authorized sender(s) to a possibly large number of authorized receivers is required. One such class of applications is the class of multicast stream applications.

Streams of data are bit sequences of a finite, but a priori unknown, length that a sender sends to one or more recipients. They occur naturally when the buffer/memory is shorter than the message, or when real-time processing is required. Digitalized audio and video are the most common multicast stream applications. However, streams are quite common in financial applications as well. Whether it be stock quotes, customer related data or other market data feeds, the volumes of this data are growing rapidly, and the data takes the form of continuous data streams rather than finite stored data sets.

* This work was supported in part by the National Science Foundation under the grant ANI-0087641.

E. Biham and A.M. Youssef (Eds.): SAC 2006, LNCS 4356, pp. 342–357, 2007.

The problems of stream authentication and stream signing have been extensively studied in the past years. Gennaro and Rohatgi [11] have proposed a stream signing scheme based on a chain of one-time signatures. A similar scheme has been presented by Zhang [28] for authentication in routing protocols. Various schemes were proposed subsequently [8,1,27,2,23,6,24] culminating with the recent adoption of TESLA as an Internet standard [19]. TESLA is also a basis for other Internet drafts (e.g., [7]), and its security and efficiency analysis can be found in [16,17,18,20].

The goal of this paper is to point out some flaws in the specification and security analysis of TESLA. Although the basic design principles of TESLA are not flawed, the suggested security assumptions about the underlying cryptographic do not provide provable security. Namely, we were able to construct examples of insecure TESLA implementations whose underlying building blocks satisfy the suggested security assumptions. We also show that provable security can be obtained by using stronger assumptions, and present an implementation whose security is based on a related-key model of a block cipher.

The outline of the paper is following. Some preliminaries are given in Section 2. In Section 3, we present examples of insecure TESLA constructions that are built using secure components. Sufficient conditions and a security proof are provided in Section 4. We propose an efficient implementation based on block ciphers in Section 5. The paper ends with the concluding remarks.

2 Background

2.1 TESLA

The Timed Efficient Stream Loss-tolerant Authentication scheme is a multicast stream authentication scheme proposed by Perrig et al [16]. Here, we briefly describe the mechanisms employed in TESLA to achieve loss-tolerance, fast transfer rates and dynamic packet rates.

The security of TESLA is based on the paradigm depicted in Figure 1. To authenticate the packet P_i of the stream, the sender first commits to the key value K_i by sending $H(K_i)$ in the packet P_{i-1}. The key K_i is only known to the sender, and it is used to compute a MAC on the packet P_i. After all recipients have received the packet P_i, the sender discloses the key value K_i in the packet P_{i+1}. The recipient verifies whether the received key value corresponds to the commitment and whether the MAC of the packet P_i computed using the received key value corresponds to the received MAC value. If both verifications are successful, the packet P_i is accepted as authentic. Note that P_i contains the commitment to the next key value K_{i+1}. To bootstrap the scheme, the first packet is signed using a digital signature scheme (e.g., RSA). If the packet P_{i-1} is lost, then the authenticity of the packet P_i and all subsequent packets cannot be verified since the commitment to the key K_i is lost. Similarly, if the packet P_{i+1} is lost, the authenticity of the packet P_i and all subsequent packets cannot be verified since the key K_i is lost.

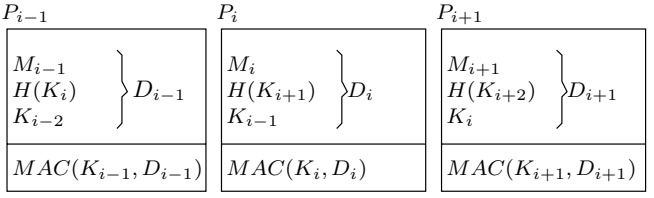

Fig. 1. The basic stream authentication scheme

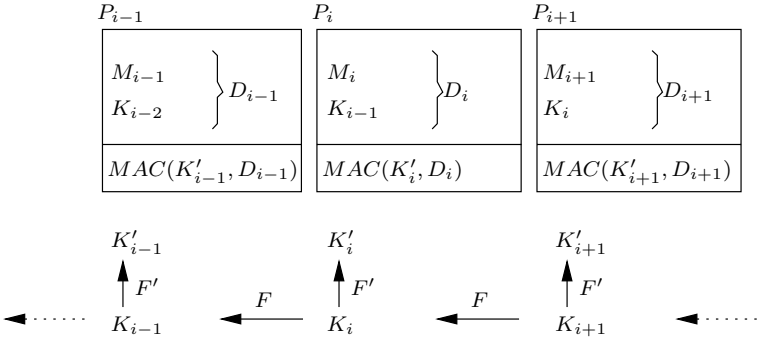

Fig. 2. TESLA Scheme II: Tolerating packet loss

Perrig et al. [16] proposed a solution to the above problem by generating the sequence of keys K_i using iterative application of a pseudo-random function to some initial value as illustrated in Figure 2. Let us denote v consecutive applications of the pseudo-random function F as $F^v(x) = F^{v-1}(F(x))$, and let $F^0(x) = x$. The sender has to pick randomly some initial key value K_n and to pre-compute n key values $K_0, \ldots, K_{n-1}$, where $K_i = F^{n-i}(K_n), i = 0, \ldots, n$. The sequence of key values is called a *key chain*. The key K'_i, which is used to authenticate the packet P_i, is derived from the corresponding key value K_i by applying the function F'. Since F is easy to compute and hard to invert, given K_i the attacker cannot compute any K_j for $j > i$. However, the recipient can compute any key value K_j from the received key value K_i, where $j < i$. Therefore, if the recipient has received a packet P_i, any subsequently received packet P_j $(j > i)$ will allow computation of $K'_i = F'(K_i)$ and verification of the authenticity of the packet P_i.

The authors suggest the function F to be implemented as $F(K_i) = f_{K_i}(0)$, where f is a target collision resistant pseudorandom function. There are no requirements imposed on the function F' in the original description of TESLA [16]. However, RFC4082 requires $F'(K_i)$ to be computed as $F'(K_i) = f'_{K_i}(1)$, where f' is a pseudorandom function. We are going to consider two cases:

– F' is an identity map. There are two main reasons why we consider this case. First, the authors make the same assumption when proving the security of TESLA in [16]. The presented proof is the only security proof of TESLA

provided by the authors. Second, if F' is an identity map or some other simple transformation, then the scheme is more efficient (i.e., we can avoid an extra PRF evaluation per packet). As shown in Section 4, the scheme can be secure even if F' is an identity map.
- $F'(K_i) = f'_{K_i}(1)$, where f' is a pseudorandom function. This is required in RFC4082.

The security of the scheme is based on the assumption that the receiver can decide whether a given packet arrived safely (i.e., before the corresponding key disclosure packet was sent by the sender). The unsafe packets are dropped. This condition severely limits the transmission rate since P_{i+1} can only be sent after every receiver has received P_i. Perrig et al [16] (TESLA Scheme III) solve this problem by disclosing the key K_i of the data packet P_i in a later packet P_{i+d}, instead of in the next packet. Another assumption made in the scheme depicted in Figure 2 is that the packet schedule is fixed or predictable, with each recipient knowing the exact sending time of each packet. This significantly restricts the flexibility of the senders. The proposed solution to this problem of dynamic packet rates is to pick the MAC key and the disclosed key in each packet only on a time interval basis. Namely, all packets sent in an interval i are authenticated using a key K_i and disclose the key K_{i-d}. This final version (TESLA Scheme IV) is the one adopted as an Internet standard. See [16,17,18,19,20] for more details.

2.2 Claimed Security of TESLA

The following theorem was given in [16].

Theorem 1. *Assume that the PRF, the MAC and the signing schemes in use are secure, and that the PRF has Target Collision Resistance property. Then, TESLA (Scheme IV) is a secure stream authentication scheme.*

To avoid complexity, the authors provide proof only for a special case when the MAC and the PRF are realized by the same function family. In their implementation, this family is the family defined by HMAC [10] when used in conjunction with MD5 [22]. However, the theorem does not require the MAC and the PRF to be realized by the same function family. We will show that the theorem does not hold in the case when the PRF and the MAC can be realized by different function families (i.e., we will disprove the theorem). Furthermore, in their proof, the authors assume that the function F' is an identity mapping. This is not the case in the RFC4082 version. Hence, their analysis does not apply to the Internet standard.

2.3 OMAC

OMAC [13] is a proven secure CBC MAC scheme that uses only one key. The evaluation of the authentication tags in OMAC is illustrated in Figure 3. The first block of the message is encrypted using a block cipher. The result is XORed with the second block and encrypted, etc. If the length of the last chunk of the message is equal to the block length n, then the last block is XORed with $L \cdot u$

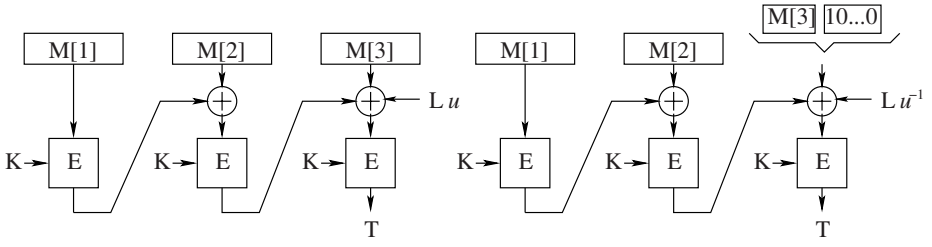

Fig. 3. One-key CBC MAC

before encryption. If the length of the last chunk of the message is less than the block length n, then 10^i padding ($i = n - 1 - |M| \bmod n$) is appended and the last block is XORed with $L \cdot u^{-1}$ before encryption. The parameter u is some known constant in $\mathrm{GF}(2^n)$, and $L = E_K(0^n)$ is an encryption of 0.

Let l be the key length. It was shown in [13] that if the function family $\{E_K\}_{K\in\{0,1\}^l}$ block cipher is a pseudorandom permutation family, then $\{\mathrm{OMAC}_K\}_{K\in\{0,1\}^l}$ (the function family defined by OMAC) is a pseudorandom function family and the OMAC scheme is unforgeable.

3 Insecure TESLA Implementations Based on Secure Components

In this section, we show that the suggested assumptions about the building blocks of TESLA are not sufficient by providing examples of insecure TESLA constructions from components that satisfy those assumptions.

3.1 Permuted-Input OMAC

In order to "break" TESLA Scheme II, we introduce Permuted-input OMAC (POMAC) scheme. The scheme will be used to authenticate the packets of the stream in our insecure TESLA Scheme II implementation. It is depicted in Fig. 4. If the length of the message m is not greater than the block size n, then the authentication tag is computed as $\mathrm{OMAC}_K(m)$. Otherwise, the message m is rotated right by n bits to derive a new message m', and the authentication tag is computed as $\mathrm{OMAC}_K(m')$.

The unforgeability of POMAC trivially follows from the unforgeability of OMAC.

Lemma 1. *Suppose that:*

- *h is a collision resistant function (i.e., it is hard to find m_1 and $m_2 \neq m_1$ s.t. $h(m_1) = h(m_2)$), and*
- *$\{f_K\}_{K\in\{0,1\}^l}$ is a function family corresponding to an unforgeable MAC scheme.*

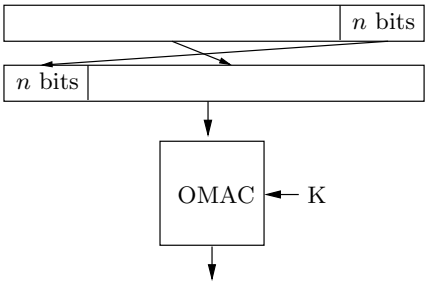

Fig. 4. Permuted-input OMAC

Then, the MAC scheme defined by the function family $\{f_K \circ h\}_{K\in\{0,1\}^l}$ *is unforgeable too.*

Proof. Assume that there is an adversary that can output a pair (m, a) where a is a valid authentication tag for a message m that hasn't been signed before. Since h is collision resistant, the hash value $h(m)$ must be different from the hash values of the previously signed messages. Hence, $(h(m), a)$ is a forgery for the MAC scheme defined by the function family $\{f_K\}_{K\in\{0,1\}^l}$. This contradicts our assumption that f is unforgeable. ■

Corollary 1. *If the function family* $\{E_K\}_{K\in\{0,1\}^l}$ *defined by the underlying block cipher is a pseudorandom permutation family, then POMAC is unforgeable.*

Proof. Follows from Lemma 1 and the facts that the initial permutation in POMAC is a bijection (i.e., collision resistant) and OMAC is unforgeable when the underlying block cipher is a pseudorandom permutation. ■

3.2 The Case When F' Is an Identity Mapping

In this section, we provide an example of an insecure TESLA construction from secure components in the case when the function F' is an identity mapping.

Suppose that the function family $\{E_K\}_{K\in\{0,1\}^n}$ is a target collision resistant pseudorandom permutation family whose members are defined on the set $\{0,1\}^n$. Note that the length of the key is equal to the block size n. AES-128 [9] is a possible candidate. Since $\{E_K\}_{K\in\{0,1\}^n}$ is a pseudorandom permutation family, it is also a pseudorandom function family (see Proposition 3.7.3 in [12]). We will use the pseudorandom permutation E_K to generate the authentication keys as illustrated in Figure 5. The key $K_{i-1} = E_{K_i}(0^n)$ is generated by encrypting 0 using the key K_i as suggested in [16]. The MAC scheme that we use in our construction is POMAC. To encrypt the message blocks in POMAC, we use the pseudorandom permutation E_K.

The PRF and the MAC as defined above satisfy the security requirements of Theorem 1. However, the resulting stream authentication scheme is not secure. Figure 5 depicts an attack on our TESLA Scheme II example by replacing the

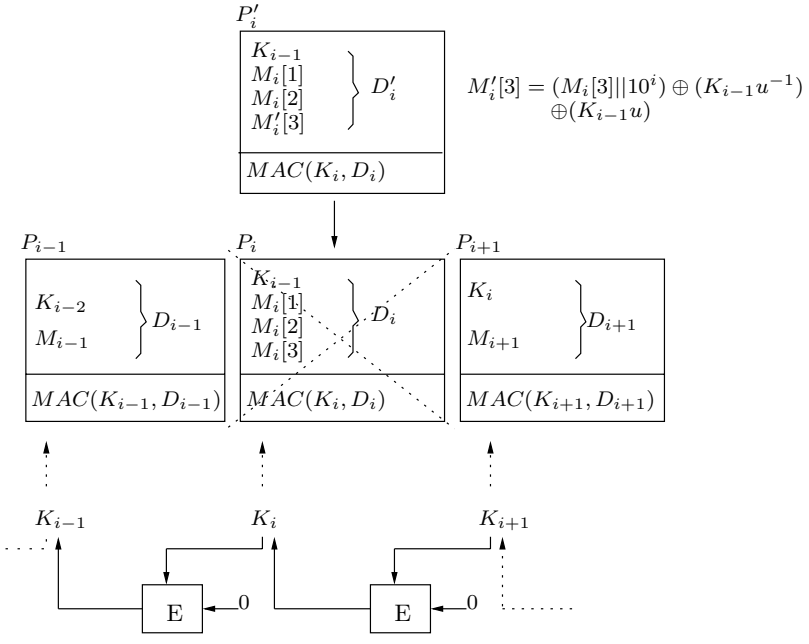

Fig. 5. Insecure TESLA implementation. MACs are computed using POMAC.

packet P_i with a packet P'_i. Without loss of generality, we assume that the message M_i consists of three chunks $M_i[1], M_i[2]$ and $M_i[3]$. The length of $M_i[1]$ and $M_i[2]$ is equal to the block length n, and the length of $M_i[3]$ is less than the block length n. This implies that $M_i[3]$ is 10^i padded and XORed with $L \cdot u^{-1}$ when computing the MAC for P_i. The forged packet P'_i is constructed by replacing $M_i[3]$ with

$$M'_i[3] = (M_i[3]||10^i) \oplus (K_{i-1} \cdot u^{-1}) \oplus (K_{i-1} \cdot u).$$

Using the equations

$$(M_i[3]||10^i) \oplus (K_{i-1} \cdot u^{-1}) = M'_i[3] \oplus (K_{i-1} \cdot u)$$

and

$$L = E_{K_i}(0^n) = K_{i-1},$$

one can easily verify that

$$\mathrm{POMAC}(K_i, D_i) = \mathrm{POMAC}(K_i, D'_i).$$

Note that all we need to compute P'_i is the key K_{i-1} and the message M_i. Since both the key K_{i-1} and the message M_i are disclosed in the packet P_i, we can compute P'_i before the key K_i is disclosed. Hence, we have succeeded in constructing a forgery for TESLA Scheme II.

3.3 Cryptanalysis of TESLA Scheme IV

As we mentioned earlier, the goal of upgrading TESLA Scheme II to TESLA Scheme IV was to achieve fast transfer rates and dynamic packet rates. The security of the upgraded scheme relies on the same principles as Scheme II, and the attack depicted in Figure 5 can be easily extended to the upgraded scheme. Moreover, the attack works with OMAC instead of POMAC as explained below in more detail.

There are two differences between Scheme II and Scheme IV that are relevant to our discussion. First, in Scheme IV, the same key is used to authenticate more than one packet sent to a given recipient. Second, the key K_{i-1} is revealed after the time interval i (assuming that the delay d is greater than one). However, note that the adversary can discard all but one packet in some time interval i, and then delay that packet so that the recipient gets the packet after the disclosure of K_{i-1}, but before the disclosure of K_i (i.e., the packet will be safe). Since, the adversary knows the value of K_{i-1} before handing the packet to the recipient, he can replace it with a forged one as in Figure 5.

The attack will work with OMAC instead of POMAC for the following reasons. The introduction of the POMAC scheme was motivated by the order of the message M_i and the key K_{i-1} within the packet P_i (Fig. 2). The initial permutation of POMAC swaps the message and the key so that the last block of D_i is a message block. In TESLA Scheme IV, the format of the packets is $P_j = \langle M_j, i, K_{i-d}, MAC(K'_i, M_j)\rangle$, where i is the interval during which the packet P_j was sent. Note that the MACs are computed over the messages M_j only, and the attack would work when OMAC instead of POMAC is used to compute the MACs. Hence, our analysis shows not only that the assumptions about the security properties of the building blocks of TESLA are not sufficient, but also that it is not unrealistic to expect that TESLA Scheme IV might be implemented insecurely.

3.4 The Case When F' Is Implemented Using a PRF

RFC4082 requires the function F to be implemented as $F(K) = f_K(0)$ (Section 3.2 of [19]), and F' to be implemented as $F'(K) = f'_K(1)$ (Section 3.4 of [19]), where f and f' are pseudorandom functions.

Although it seems that the scheme is secure when f and f' are identical[1], the RFC does not require f and f' to be identical. On the contrary, the use of different symbols to denote them suggests that they can be different. In this case, the new TESLA Scheme II still suffers from the flaw discussed in Section 3.2. Namely, we can view f' as a part of the key scheduling algorithm of the underlying block cipher. The function F of the insecure TESLA construction is now implemented as $F(K_i) = E_{f'_{K_i}(1)}(0)$ (see Figure 6). It is clear that $K_{i-1} = F(K_i)$ leaks the encryption of zero since $K_{i-1} = E_{K'_i}(0)$, and we can mount the same attack.

In addition to the old flaw, the modification of the scheme introduces a new one. Consider the following "naive" implementation. The function F is implemented

[1] The reader should be aware that there is no security proof provided for this case.

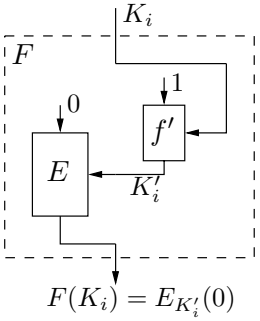

Fig. 6. The function F leaks the encryption of zero $E_{K'_i}(0)$

as $F(K_i) = f_{K_i}(0)$, where f is a target collision resistant pseudorandom function family. The function F' is implemented as $F'(K_i) = f'_{K_i}(1)$, where $f'_{K_i}(x) = f_{K_i}(x-1)$. One can easily show that f' is a pseudorandom function. It is not hard to verify that the commitment $F(K_i)$ discloses the authentication key K'_i: $F(K_i) = f_{K_i}(0) = f_{K_i}(1-1) = f'_{K_i}(1) = K'_i$. Although this implementation is very unlikely, it demonstrates the threat of exploiting the knowledge of the commitment $F(K_i)$ to compute the authentication key K'_i.

3.5 Cryptanalysis of the RFC4082 TESLA Version

The analysis presented in Section 3.4 can be extended to the TESLA version described in RFC4082. We use the same arguments as in Section 3.3. The safe packet test only checks whether a packet authenticated using a key K_i was received before the disclosure of the key K_i. Hence, the adversary can delay the packet until the key K_{i-1} is disclosed, and then replace it with a forged one. The aforementioned security flaws cannot be patched by simply modifying the safe packet test so that the receiver checks whether the packet was received before the disclosure of the key value K_{i-1}. In this case, the adversary might be able to use $K_{i-2} = F(F(K_i))$ or some previous key value to mount an attack.

4 Sufficient Assumptions About the Components of TESLA

The attacks on the insecure implementations that were presented in Section 3 are based on the following observation. The security of the MAC scheme that is used to authenticate the packets is proven in a setting where the adversary has access to a signing oracle and a verifying oracle. In the case of TESLA, we have a different setting. Now, the adversary has access to an additional oracle that computes the commitment $F(K)$ to the secret key K which is used by the MAC scheme. The adversary can exploit the knowledge of $F(K)$ to construct a forgery.

It is clear from the discussion above that we need to make an additional assumption about the function F and the MAC scheme. Namely, the MAC scheme must remain secure even when the commitment of the secret key used by the MAC scheme is revealed.

Definition 1. *A MAC scheme is* known F-commitment unforgeable *if there is no efficient adversary that given a commitment $F(K)$ of the secret key that is in use can break the MAC scheme with non-negligible probability.*

An example, which demonstrates that one can achieve known F-commitment unforgeability, is provided in Section 5.2.

We also make the following minor modification of TESLA Scheme II. Each time a stream is authenticated, the sender selects a unique number N_s (e.g., using a counter) which is securely communicated to the recipients. The number N_s is included as part of the authenticated data in each packet of the stream including the bootstrap packet. So, we assume that the format of the messages is $M_i = \langle N_s, i, C_i \rangle$, where C_i is the actual chunk of the stream [2] [3].

The following theorem holds for the security of the slightly modified TESLA Scheme II.

Theorem 2. *Suppose that:*

1. *the digital signature scheme, which is used to bootstrap TESLA, is unforgeable,*
2. *the function $F(K) = f_K(0)$, where f is a pseudorandom function, is collision resistant,*
3. *the MAC scheme, which is used to authenticate the chunks of the stream, is known F-commitment unforgeable, and*
4. *F' is an identity mapping.*

Then, TESLA Scheme II is a secure multicast stream authentication scheme.

The proof is given in Appendix A.

Note that F' is an identity mapping, while in RFC4082, F' is realized using a pseudorandom function. Hence, the scheme that is analyzed here is somewhat more efficient than the Internet standard.

The requirement for collision resistance of the function F can be slightly weakened. Assuming that there is a bound on the number of packets within a stream, it is not hard to show that TESLA Scheme II is secure when the function F is collision resistant in the following sense: Given a randomly selected value K and a bound $L \geq 1$, it is hard to find K' and a positive integer $l \leq L$ such that $F^l(K) = F(K')$ and $F^{l-1}(K) \neq K'$. A function that satisfies the aforementioned property is said to be *bounded iteration collision resistant.*

[2] TESLA does not provide ordering of the packets that are authenticated using the same key. We use sequence numbers to prevent malicious reordering of the packets.

[3] To reduce the communication overhead one can communicate N_s only once, and then just use it to compute the signature and the MACs.

5 A Candidate Implementation of TESLA

In this section, we propose an implementation that uses block ciphers to realize the different components of TESLA.

5.1 CKDA-PRPs

When cryptanalyzed, block ciphers are not considered secure unless they are resistant to related-key attacks [4]. A theoretical treatment of block ciphers resistant to related-key attacks was given in [3], where it was shown that under some restrictions one can achieve resistance to related-key attacks. We are going to use a model of a more specific case: the adversary can query oracles that use keys whose difference was chosen by the adversary (e.g., related-key differential cryptanalysis [15,14]).

We define a CKDA secure (i.e., secure against Chosen Key Difference Attacks) pseudorandom permutation family as a pseudorandom permutation family such that one cannot tell apart a pair of permutations randomly selected from the family and a pair of permutations from the family whose index (key) difference is $c \neq 0$, where c is selected by the adversary. A CKD test is a Turing machine A with access to four oracles $\mathcal{E}_1$, $\mathcal{D}_1$, $\mathcal{E}_2$ and $\mathcal{D}_2$. A selects a non-zero l-bit string c. The oracle $\mathcal{E}_1$ is selected to be a random permutation E_K from the permutation family $\{E_K\}_{K\in\{0,1\}^l}$, and the oracle $\mathcal{D}_1$ is selected to be its inverse. According to a secret random bit b, the oracle $\mathcal{E}_2$ is selected to be either the permutation $E_{K\oplus c}$ or a random permutation $E_{K\oplus r}$, where c is the public non-zero constant and r is a random bit string of length l (i.e., $K \oplus r$ is random and not related to K). The oracle $\mathcal{D}_2$ computes the inverse of $\mathcal{E}_2$. The algorithm A outputs 0 or 1. The advantage of the CKD test is defined as

$$\mathrm{Adv}_A((E_K, E_{K\oplus c}), (E_K, E_{K\oplus r})) = \frac{1}{2}(E[A^C] - E[A^R])$$

where $E[A^C]$ (resp., $E[A^R]$) is the probability that A will output 1 when the difference between the secret keys is a known non-zero constant (resp., random l-bit string).

Definition 2. *The pseudorandom permutation family $\{E_K\}_{K\in\{0,1\}^l}$ is a $[t, q, \epsilon]$-*secure CKDA pseudorandom permutation family (or $[t, q, \epsilon]$-secure CKDA-PRP) *if there is no CKD test that runs in at most t time, sends at most q queries to the oracles and has at least ϵ advantage.*

5.2 TESLA Implementation Via CKDA-PRPs

The following theorem provides a function F and a MAC scheme such that the MAC scheme is known F-commitment unforgeable.

Theorem 3. *Let the function family $\{E_K\}_{K\in\{0,1\}^n}$ corresponding to the block cipher used by OMAC be CKDA secure. Let $F : \{0,1\}^n \to \{0,1\}^n$ be defined as $F(K) = E_{K\oplus c}(0)$, where $c = 0^{n-1}1$. Then, OMAC is a known F-commitment unforgeable MAC scheme.*

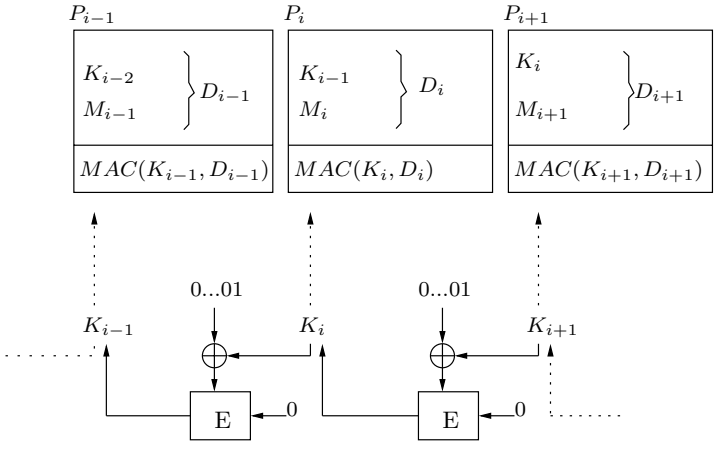

Fig. 7. TESLA implementation using a block cipher resistant to related-key cryptanalysis

Proof. An adversary A_1 that given a commitment $F(K')$ to some randomly selected key K' can break OMAC with probability ϵ can be easily converted into an adversary A_2 that can break OMAC with the same probability. In particular, A_2 can randomly select the key value K' and submit the commitment $F(K')$ to A_1. A_1's output will be A_2's output. Since OMAC is unforgeable, there is no adversary that can break OMAC with significant probability given a commitment to a randomly select key.

Now, assume that there is an adversary A_3 that can break OMAC given the commitment $F(K)$ to the secret key K that is in use. We can construct a CKD test as follows. We run the adversary A_3 and answer its queries by querying the oracles $\mathcal{E}_1$ and $\mathcal{E}_2$. If A_3 manages to produce a forgery we output 1, otherwise we output 0. Obviously, the advantage of the CKD test will be significant since the probability $E[A^C]$ is significant (OMAC is not known F-commitment unforgeable) and the probability $E[A^R]$ is small (OMAC is unforgeable). ■

The implementation that we propose here is depicted in Figure 7. It is similar to the insecure implementation shown in Figure 5. The only difference is that the key value K_{i-1} is derived by encrypting zero using the key $K_i \oplus 0^{n-1}1$ instead of the key K_i. A similar secure variant of the insecure implementation can be obtained by using a function F' that derives the key K'_i by flipping the last bit of K_i instead of using an identity map. We must note that the security of the proposed scheme is also based on the assumption that $h(x) = E_x(0)$ is collision resistant (one of the assumptions made in Theorem 2). While there are some constructions and possibility results regarding hash functions based on block ciphers [21,5], we are not aware of any results regarding the collision resistance of $h(x) = E_x(0)$ where E is some widely used cipher with relatively large block size (e.g., AES).

6 Conclusion

We have shown that the assumptions about the components of TESLA are not sufficient and can potentially lead to insecure implementations. We also provided sufficient conditions for the security of the scheme and proposed an implementation based on block ciphers.

References

1. Anderson, R., Bergadano, F., Crispo, B., Lee, J., Manifavas, C., Needham, R.: A New Family of Authentication Protocols. ACM Operating Systems Review 32(4), 9–20 (1998)
2. Bergadano, F., Cavagnino, D., Crispo, B.: Chained Stream Authentication. In: Proceedings of Selected Areas in Cryptography 2000, pp. 142–155 (2000)
3. Bellare, M., Kohno, T.: A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In: Biham, E. (ed.) Advances in Cryptology – EUROCRPYT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)
4. Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. Journal of Cryptology 7(4), 229–246 (1994)
5. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 225–320. Springer, Heidelberg (2002)
6. Canneti, R., Garay, J., Itkis, G., Micciancio, D., Naor, M., Pinkas, B.: Multicast security: A taxonomy and some efficient constructions. In: Infocom '99 (1999)
7. Carrara, E., Baugher, M.: The Use of TESLA in SRTP. Internet draft, `http://ietfreport.isoc.org/ids-wg-msec.html`
8. Cheung, S.: An Efficient Message Authentication Scheme for Link State Routing. In: Proceedings of the 13th Annual Computer Security Application Conference (1997)
9. FIPS PUB 197, The Advanced Encryption Standard
10. FIPS PUB 198, The Keyed-Hash Message Authentication Code (HMAC)
11. Gennaro, R., Rohatgi, P.: How to Sign Digital Streams. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 180–197. Springer, Heidelberg (1997)
12. Goldreich, O.: Foundations of Cryptography. Cambridge University Press, Cambridge (2001)
13. Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
14. Jakimoski, G., Desmedt, Y.: Related-key Differential Cryptanalysis of 192-bit Key AES Variants. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 208–221. Springer, Heidelberg (2004)
15. Kelsey, J., Schneier, B., Wagner, D.: Related-key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2 and TEA. In: Proceedings of ICICS'97, pp. 233–246. Springer, Heidelberg (1997)
16. Perrig, A., Canneti, R., Tygar, J.D., Song, D.: Efficient Authentication and Signing of Multicast Streams Over Lossy Channels. In: Proceedings of the IEEE Security and Privacy Symposium (2000)
17. Perrig, A., Canneti, R., Song, D., Tygar, J.D.: Efficient and Secure Source Authentication for Multicast. In: Proceedings of the Network and Distributed System Security Symposium (2001)

18. Perrig, A., Canneti, R., Tygar, J.D., Song, D.: The TESLA Broadcast Authentication Protocol. RSA CryptoBytes 5(2) (2002)
19. Perrig, A., Song, D., Canneti, R., Tygar, J.D., Briscoe, B.: Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction. Internet Request for Comments, RFC 4082 (June, 2005)
20. Perrig, A., Tygar, J.D.: Secure Broadcast Communication in Wired and Wireless Networks. Kluwer Academic Publishers, Dordrecht (2002)
21. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, Springer, Heidelberg (1994)
22. Rivest, R.L.: The MD5 message digest algorithm. Internet Request for Comments, RFC 1321 (April 1992)
23. Rohatgi, P.: A compact and fast hybrid signature scheme for multicast packet authentication. In: 6th ACM Conference on Computer and Communications Security, November 1999 (1999)
24. Syverson, P.F., Stubblebine, S.G., Goldschlag, D.M.: Unlinkable serial transactions. In: FC 1997. LNCS, vol. 1318, Springer, Heidelberg (1997)
25. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis for Hash Functions MD4 and RIPEMD. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
26. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
27. Wong, C.K., Lam, S.S.: Digital Signatures for Flows and Multicasts. In: Proceedings of IEEE ICNP '98 (1998)
28. Zhang, K.: Efficient Protocols for Signing Routing Messages. In: Proceedings of the Symposium on Network and Distributed System Security (1998)

A Proof of Theorem 2

Assume that the adversary can break the stream authentication scheme. In other words, the adversary in cooperation with some of the recipients can trick another recipient u to accept a forged packet of the stream as valid.

Let i be the smallest integer such that the contents D'_i is accepted as valid by the recipient u when the original contents D_i is different from D'_i. There are three possible events:

Event 1. If i is zero, then the adversary has managed to forge the bootstrap packet which was signed using a digital signature scheme.

Event 2. If i is greater than zero and the key that u used to verify the validity of D'_i is equal to the original key K_i, then the adversary has managed to produce a forgery for the message authentication scheme due to the uniqueness of $\langle N_s, i\rangle$.

Event 3. If i is greater than zero and the key K_i^f that u used to verify the validity of D'_i is different than the original key K_i, then the adversary has managed to find a collision for the function F. Let $K_i^f, K_{i-1}^f = F(K_i^f), \ldots$ be a key chain derived from K_i^f, and let $K_i, K_{i-1} = F(K_i), \ldots$ be a key

chain derived from K_i. The user u verified the validity of the key value K_i^f by checking whether $F^l(K_i^f)$ is equal to some previously authenticated key value K_{i-l}^f. Since i is the smallest index of a packet whose contents D_i' is different from the original contents D_i, the received key value K_{i-l}^f must be equal to the original key value $K_{i-l} = F^l(K_i)$. Hence, there is an index $i - l \leq j < i$ s.t. $K_{j+1} \neq K'_{j+1}$ and $F(K_{j+1}) = K_j = K'_j = F(K'_{j+1})$.

Given an efficient adversary A_{SA} that breaks the stream authentication scheme with significant probability, we will construct an adversary A_{S} for the signature scheme, an adversary A_{MAC} for the MAC scheme and an adversary A_{F} for the function F, and show that at least one of these adversaries has significant success probability. All three adversaries simulate the network using sets of read and write tapes for the users and for the adversary. They differ in the following aspects:

1. The adversary for the signature scheme answers the stream signing queries by randomly selecting initial key values, computing the key chains and using the signing oracle for the bootstrap packets. Whenever A_{SA} manages to forge a bootstrap packet, A_{S} outputs the forged message/signature pair. Otherwise, it outputs a randomly selected message/signature pair.
2. The adversary for the MAC scheme guesses which stream will be forged and what will be the smallest index i of a forged packet within the stream. If the guess is that the stream will not be forged, then A_{MAC} answers the stream signing query by randomly selecting the initial key value. Otherwise, the adversary uses the given value $K_{i-1} = F(K_i)$ to derive the keys that will be used to authenticate the packets $P_1, \ldots P_{i-1}$, and computes the MAC for the packet P_i by submitting a query to the (MAC) signing oracle. If the adversary for the stream scheme manages to forge the i-th packet, then the adversary for the MAC scheme outputs the forged message/MAC pair. Otherwise, it outputs a randomly selected message/MAC pair.
3. The adversary for the function F answers the stream signing queries by randomly selecting initial key values, computing the key chains and using a private key when signing the bootstrap packets. In the case when Event 3 occurs, A_{F} finds and outputs a pair of key values that collide. Otherwise, it outputs two randomly selected key values.

It is easy to show that if the probabilities of Event 1 and Event 3 are significant, then the success probabilities of the corresponding adversaries A_{S} and A_{F} are significant too. To derive a relation between the probability of Event 2 and A_{MAC}, we need the following Lemma.

Lemma 2. *If f is a pseudorandom function, then there is no efficient algorithm that can distinguish between a random key value and the key value $F^l(K)$ derived from a secret random key K by $l \geq 1$ iterations of the function F.*

Proof. We can prove the Lemma by induction. If there is an algorithm that can tell apart between $F(K) = f_K(0)$ and a random key value, then we can construct an algorithm that can distinguish between the function family $\{f_K\}$ defined by

f and the random function family. Now, assume that there is no algorithm that can tell apart between the key $K_{l-1} = F^{l-1}(K)$ and a random key value. Since the function f and the key K_{l-1} are pseudorandom, the key $K_l = f_{K_{l-1}}(0)$ will be indistinguishable from a random key too. ■

Assume that n_s and L are the maximum number of streams and the maximum number of packets within a single stream respectively. The probability that A_{MAC} will guess the forged stream and the index i of the first forged packet within the stream is $\frac{1}{n_s L}$. According to Lemma 2, there is no efficient algorithm that can distinguish with significant probability between the secret key K_i used by the MAC scheme and a key that is derived from some initial key value by $l-i$ iterations of the function F. Hence, if the probability ϵ of Event 2 is significant, then the success probability of A_{MAC} will be approximately $\frac{\epsilon}{n_s L}$.