CRYPTANALYSIS OF A ROBUST KEY AGREEMENT BASED ON PUBLIC KEY AUTHENTICATION[*]

Mohsen Toorani[†]

This paper considers security analysis of the YAK, a public key-based authenticated key agreement protocol. The YAK protocol is a variant of the two-pass HMQV protocol, but uses zero-knowledge proofs for proving knowledge of ephemeral values. In this paper, we show that the YAK protocol lacks joint key control and perfect forward secrecy attributes, and is vulnerable to some attacks including unknown key-share and key-replication attacks. This invalidates the semantic security of the protocol in several security models. There are also other considerations regarding the impersonation and small subgroup attacks.

Keywords: Public key cryptography; Cryptographic protocols; Authenticated key exchange; Security models; Key control.

[*]This is a preprint which is not the same as the finally published manuscript. Final paper will appear in *Journal of Security and Communication Networks* and can be accessed at dx.doi.org/10.1002/sec.1373. All rights reserved ©2015 John Wiley & Sons.

[†]Department of Informatics, University of Bergen, Norway.

1 Introduction

Authenticated key exchange (AKE) protocols are a well-known and hot topic in information security. The goal is to establish a cryptographic session key between legitimate entities in an authenticated manner. AKE protocols use a combination of some secret parameters and random challenge-responses. Many AKE protocols have been proposed in the literature, but some of them have been shown to have security problems [1–5].

Several security models have been developed for AKE protocols. Each of them has a different assumption about capabilities of the adversary. The BR93 [6], CK01 [7], and eCK [8] are some of the well-known security models for AKE protocols. A protocol that is proved to be secure in a security model might be insecure in certain other security models, due to hierarchy of assumptions on capabilities of adversaries and valid attacks. Several security properties must be satisfied by AKE protocols, and they should obviously withstand well-known attacks. In addition to implicit key authentication and key confirmation, it is desirable for AKE protocols to provide known-key security, forward secrecy, and resilience to well-known attacks such as key-compromise impersonation (KCI), unknown key-share (UKS), replay, and Denning-Sacco attacks [9–11]. Key agreement protocols should also provide joint key control [12].

The YAK protocol [13, 14] is proposed as an improvement to the two-pass HMQV protocol [1]. The goal was to thwart a hypothetical attack on self-communication mode where two communicating entities use the same certificate and public keys. The YAK protocol removes the XCR signatures for the HMQV protocol, but adds zero-knowledge proofs (ZKP) for proving knowledge of ephemeral secret keys at both parties. It is based on public key cryptography, and requires a public key infrastructure (PKI) for certifying public keys. In [13, 14], there are claims for security and efficiency of the YAK protocol, but there is no formal proof of security. In this paper, we analyze the security of the YAK protocol, and show that it is vulnerable to several attacks including unknown key-share (UKS), and key-replication attacks. This invalidates the semantic security of the protocol in many security models. There are also other considerations regarding the impersonation and small subgroup attacks. Although the YAK protocol is called an AKE protocol, the authentication is just zero-knowledge proof of a random number, generated by the other party. A key confirmation is then a must to avoid an impersonation attack. In this case, it is necessary to check the prime order of public keys in order to avoid the small subgroup attacks. We also show that the YAK protocol lacks the joint key control and perfect forward secrecy (PFS) attributes.

2 Security models for AKE protocols

Many security models have been developed for security analysis of AKE protocols. The BR93 security model [6] was the first game-based model in this context. The BR95 [15], BPR [16], and CK01 [7] models are other security models, inspired by the BR93 model. The AKE security experiment in the CK01 model involves some honest parties and an adversary $\mathcal{M}$ connected through an unauthenticated network. $\mathcal{M}$ selects parties to execute key exchange sessions, and selects an order in which the sessions will be executed. $\mathcal{M}$ is allowed to take full control of any party (a *Corrupt* query), to reveal the session key of any session (a *Reveal* query), or to reveal session-specific secret information of any session (a *Session-State Reveal* query). State reveal queries are motivated by practical scenarios where an adversary can gain access to the ephemeral data. The *Session-State Reveal* query in the CK01 model allows $\mathcal{M}$ to have the state-specific information of the parties without revealing the long-term secret information. Then, there is no provision for some practical attacks including the KCI attack. Resilience to the KCI attack is an important security attribute for AKE protocols. If the private key of an entity $\mathcal{A}$ is compromised, an adversary $\mathcal{M}$ can impersonate $\mathcal{A}$ in one-factor authentication protocols. However, such compromise should not enable $\mathcal{M}$ to impersonate other honest entities in communication with $\mathcal{A}$ [17]. Such provisions are considered in the eCK (extended-CK) model [8]. The party executing the session is called the *owner* of the session, and the other party is called the *peer*. The *matching session* to an AKE session is the corresponding AKE session which is supposed to be executed by the peer with the owner. Definition of the matching session may differ in different security models.

The eCK model is intended to be stronger than the CK01 model. However, they are incomparable [18, 19]. The eCK model allows various combinations for leakage of long-term and ephemeral private keys, but not both leakages happening at the same entity. Here are some practical attack scenarios that are not considered in the CK01 model, but they are allowed in the eCK model [8]:

- KCI attack: Adversary reveals a long-term private key of a party, and impersonates others to this party.
- Ephemeral KCI attack: An adversary reveals the ephemeral private key of a party, and impersonates others to this party.
- Two honest parties execute matching sessions. Adversary reveals the ephemeral private keys of both parties, and tries to learn the session key.
- Two honest parties execute matching sessions. Adversary reveals the ephemeral private key of one party and the long-term private key of the other party, and tries to learn the session key.
- Two honest parties execute matching sessions. Adversary reveals the long-term private keys of both parties prior to the execution of the session, and tries to learn the session key.

3 Review of the YAK protocol

The YAK protocol [13, 14] has been proposed as an improvement to the HMQV protocol [1]. The idea was to thwart an attack on the HMQV protocol in a self-communication mode. The attack was based on the assumption that an entity uses its public key and digital certificates on two devices, and uses the HMQV protocol for a key exchange between devices. Then, an attacker impersonates the other device, establishes two parallel sessions with the honest entity, and replies the message that receives from an entity in one of two sessions to another entity in the other session. The adversary cannot find the session key, but may replay a hypothetical command encrypted using the established session key. The attack was called a *wormhole attack* [13, 14], and considered as a UKS attack because an entity believes that it shared a session key with itself, while it really shared a session key with an adversary; even though the adversary cannot derive the session key. It is noteworthy that definition of the UKS attack in some security models requires that an adversary resides between two distinct entities. In that setting, the aforementioned attack does not make sense.

The YAK protocol is claimed to be the simplest public-key AKE protocol [13, 14]. Actually, it would be considered as a variant to the HMQV protocol [1] after removing the XCR signatures by setting $d = e = 1$ in the HMQV protocol, and adding ZKPs of ephemeral private keys to the protocol. Specifications of the YAK protocol include a high-level description which requires some ZKPs, and suggests using the Schnorr signature [20] for ZKPs. Figure 1 depicts the original description of the YAK protocol, as appeared in [13, 14]. The YAK protocol requires two passes of communication between two communicating entities, Alice and Bob. There is not any provision for the implicit key confirmation. Then, for having the key confirmation, extra pass(es) are required. In the rest of this paper, $\mathcal{A}$ denotes Alice, $\mathcal{B}$ denotes Bob, and $\mathcal{M}$ denotes the adversary in an active attack.

Let G denotes a subgroup of $\mathbb{Z}_p^*$ with prime order q where p is prime, $q|p-1$ to avoid the key recovery attack on DLP schemes [21], and q is big enough for intractability of the computational Diffie-Hellman (CDH) problem. Let g be a generator in G, and both $\mathcal{A}$ and $\mathcal{B}$ agree on (G, g). $\mathcal{A}$ and $\mathcal{B}$ register on a PKI, and get a certificate for their public keys from the certificate authority (CA). Private and public key of $\mathcal{A}$ is a and $PK_A = g^a$, respectively. Private and public key of $\mathcal{B}$ is b and

$$\begin{array}{ccc}
\textbf{Alice } (\hat{A}, PK_A = g^a) & & \textbf{Bob } (\hat{B}, PK_B = g^b) \\
x \in_R \mathbb{Z}_q & & \\
X = g^x & & \\
& \xrightarrow{X,\ \mathrm{KP}\{x\}} & \\
& & \text{Verify KP}\{x\} \\
& & y \in_R \mathbb{Z}_q \\
& & Y = g^y \\
& & k = H((X.PK_A)^{y+b}) = H(g^{(x+a)(y+b)}) \\
& \xleftarrow{Y,\ \mathrm{KP}\{y\}} & \\
\text{Verify KP}\{y\} & & \\
k = H((Y.PK_B)^{x+a}) = H(g^{(x+a)(y+b)}) & &
\end{array}$$

Fig. 1: *The YAK protocol [13, 14]*

$PK_B = g^b$, respectively. The YAK protocol can be described through the following steps:

1. $\mathcal{A}$ selects a random number x so that $x \in_R [0, q-1]$. $\mathcal{A}$ computes $X = g^x$, and generates zero-knowledge proof of x, denoted by $KP\{x\}$. $\mathcal{A}$ sends X and $KP\{x\}$ to $\mathcal{B}$.

2. $\mathcal{B}$ selects a random number y so that $y \in_R [0, q-1]$. $\mathcal{B}$ computes $Y = g^y$, and generates zero-knowledge proof of y, denoted by $KP\{y\}$. $\mathcal{B}$ sends Y and $KP\{y\}$ to $\mathcal{A}$.

3. $\mathcal{A}$ verifies the received $KP\{x\}$. If verified, $\mathcal{A}$ computes the session key as $k = H((Y.PK_B)^{x+a})$ in which H is a hash function.

4. $\mathcal{B}$ verifies the received $KP\{y\}$. If verified, $\mathcal{B}$ computes the session key as $k = H((X.PK_A)^{y+b})$.

By completion of successful verifications, $\mathcal{A}$ and $\mathcal{B}$ are claimed to authenticate each other, and obtain the same session key k. The correctness can be simply verified as $k = H((X.PK_A)^{y+b}) = H((Y.PK_B)^{x+a}) = H(g^{(x+a)(y+b)})$. Hence, $\mathcal{A}$ and $\mathcal{B}$ obtain the same value for the session key k.

Description of the YAK protocol, depicted in Figure 1, includes some black boxes for ZKPs and their verifications without further details.

YAK's specifications [13, 14] suggest using the Schnorr signature [20] for the ZKPs, without a clear description of the protocol.

The Schnorr signature scheme includes a secure hash function H, and is provably secure in the random oracle model. It has been used for a variety of applications in the literature [22–24]. The Schnorr identification scheme is zero-knowledge for an honest verifier and a small challenge space. The scheme would be made non-interactive through a Fiat-Shamir transformation [25], assuming that there exists a secure cryptographic hash function. To prove the knowledge of the exponent x in $X = g^x$, an entity may send $\{SignerID, OtherInfo, U = g^u, r = u - xh\}$ in which $SignerID$ is the unique identifier of the entity, $OtherInfo$ may include auxiliary information, $u \in_R \mathbb{Z}_q$, and h maybe defined as $h = H(g, U, X, SignerID, OtherInfo)$ [13, 14]. For verifying knowledge proofs for x, the verifier computes h and verifies that $U = g^r X^h$. Including $SignerID$ in calculation of h is not part of the original Schnorr scheme [20]. There is also an extra assumption in [13, 14] that a verifier in the Schnorr scheme will check that X lies in the subgroup of prime order q, although such verification is not part of the Schnorr scheme [20].

4 Security Problems of the YAK Protocol

The YAK protocol [13, 14] is called a key agreement protocol. However, as we show in this section, it lacks the joint key control attribute which is required by any key agreement protocol. This is specifically important because the HMQV protocol to some extent provides this attribute, but the YAK protocol as an improvement to the HMQV protocol fails to provide it. Furthermore, we show that the YAK protocol is vulnerable to the UKS and key-replication attacks. Such vulnerability invalidates the security of the YAK protocol in the HMQV, eCK, and other security models wherein those attacks are valid. There are also further considerations regarding impersonation and small subgroup attacks that are considered in this section. It is trivial to show that the YAK protocol

Protocol	Key derivation function
HMQV [1]	$H(g^{(x+aH_1(g^x,\hat{B}))(y+bH_1(g^y,\hat{A}))})$
FHMQV [26]	$H(g^{(x+aH_1(g^x,g^y,\hat{A},\hat{B}))(y+bH_1(g^y,g^x,\hat{A},\hat{B}))}, \hat{A}, \hat{B}, g^x, g^y)$
NAXOS [8]	$H(g^{aH_1(y,b)}, g^{bH_1(x,a)}, g^{H_1(x,a)H_1(y,b)}, \hat{A}, \hat{B})$
CMQV [27]	$H(g^{(H_1(x,a)+aH_2(g^{H_1(x,a)},\hat{A},\hat{B}))(H_1(y,b)+bH_2(g^{H_1(y,b)},\hat{A},\hat{B}))}, g^{H_1(x,a)}, g^{H_1(y,b)}, \hat{A}, \hat{B})$
YAK [13, 14]	$H(g^{(x+a)(y+b)})$

Table 1: *Session key derivation functions in different protocols*

does not provide the PFS, and is then insecure in the eCK-PFS security model.

4.1 Key control

The joint key control is a requirement for any key *agreement* protocol [9, 12]. It means all the intended participants should be involved in calculation of a session key, and neither of them can predetermine the value of the session key or enforce it to fall into a pre-determined interval. The YAK protocol lacks the joint key control because of a simplified session key derivation function. Table 1 compares the session key derivation functions of the YAK protocol and some other AKE protocols, where H, H_1, and H_2 denote hash functions. In the HMQV protocol, the session key is computed as $k = H(g^{(x+da)(y+eb)})$ where $d = H(X, \hat{B})$ and $e = H(Y, \hat{A})$. However, the session key in the YAK protocol is computed as $k = H(g^{(x+a)(y+b)})$ which makes the protocol to miss the joint key control attribute. Figure 2 depicts how $\mathcal{B}$ can enforce the established session key to be $k = H(1)$. A similar scenario can be accomplished by $\mathcal{A}$ where $\mathcal{A}$ puts $x = -a$. As an entity can enforce the outcome of the key agreement to have a predetermined value, the YAK protocol lacks the joint key control attribute.

4.2 Unknown Key-Share attack

Resilience to the UKS attack is an important security attribute for AKE protocols [9]. In a UKS attack, $\mathcal{M}$ tries to make an entity $\mathcal{A}$ believe

Alice ($\hat{A}, PK_A = g^a$)		**Bob ($\hat{B}, PK_B = g^b$)**
$x \in_R \mathbb{Z}_q$		
$X = g^x$		
	$\xrightarrow{X,\ \mathrm{KP}\{x\}}$	
		Verify $\mathrm{KP}\{x\}$
		$y = -b$
		$Y = {PK_B}^{-1}$
		$k = H(1)$
	$\xleftarrow{Y,\ \mathrm{KP}\{y\}}$	
Verify $\mathrm{KP}\{y\}$		
$k = H(1)$		

Fig. 2: *An entity can enforce the YAK protocol to establish a predetermined session key*

that the session key is shared with $\mathcal{B}$, while $\mathcal{B}$ believes that the session key is shared with $\mathcal{M}$ [28]. However, an attacker is not required to compute the session key [29]. A practical scenario where an attacker can take benefit of a UKS attack is presented in [29]. The UKS attack can be against the initiator or the responder, based on whether the entity which is misled to accept a wrong identity of the partner entity is the initiator or the responder. The UKS attack is feasible when a key exchange protocol fails to provide an authenticated binding between the session key and identifiers of the honest entities [30]. Typically, there are two kinds of UKS attacks:

1. In the first type which is referred to as a *public key substitution UKS attack*, an adversary $\mathcal{M}$ registers $\mathcal{A}$'s public key PK_A as its own public key, i.e. $PK_M = PK_A$.

2. In the second type, the adversary has a valid public-private key certified by the CA, and tries to perform a UKS attack.

As in the first type UKS attack, the adversary does not know $\mathcal{A}$'s private key, this attack can be prevented by requiring the CA to check

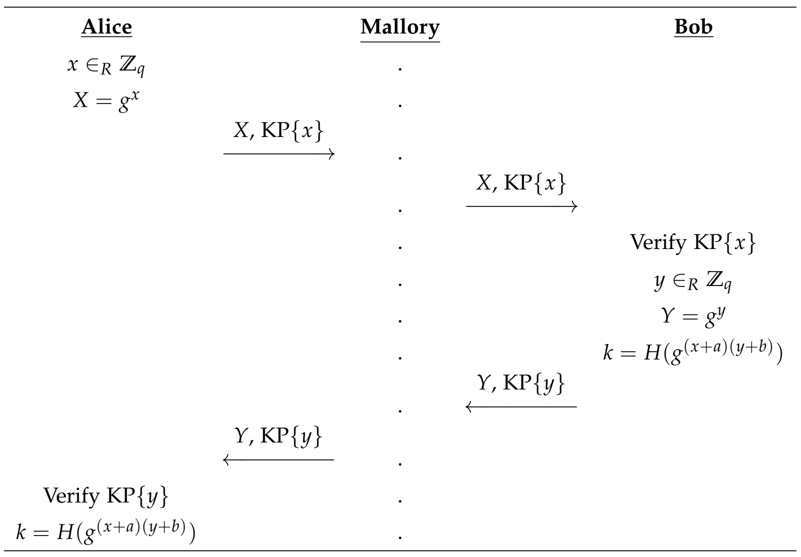

Fig. 3: *A scenario for the UKS and key-replication attacks on the YAK protocol*

during the certificate issuing procedure that each entity possesses a private key corresponding to its public key. However, this contradicts with the adversarial power in some security models for AKE protocols. For example, the eCK model allows an adversary to register arbitrary public keys for adversary-controlled parties without any check such as proof-of-possession (PoP) done by the CA [8]. This means that a public key substitution UKS attack is allowed in the eCK model. Furthermore, in more realistic models such as the ASICS model [31] which concerns the certification procedure in PKI-based protocols, an adversary can register arbitrary valid and invalid public keys at the CA via *pkregister* and *npkregister* queries, respectively. If security of a protocol depends on extra assumptions that limit the capabilities of an adversary, the security of the protocol cannot be proved in all stronger security models wherein an adversary has more adversarial power.

As depicted in Table 1, excluding the YAK protocol, the session key derivation function of all other protocols include $\hat{A}$ and $\hat{B}$. In the YAK protocol, there is no specific binding between identifiers of participants and authentication credentials, and the session key derivation function. Then, the principle description of the YAK protocol presented in Figure 1, is potentially susceptible to a UKS attack. Because of having ZKPs of ephemeral private keys at both parties, the second type UKS attack is not applicable to the YAK protocol. However, the first type UKS attack is feasible. Since the protocol is symmetric, the attack can be done on either initiator or responder. In a public key substitution UKS attack against the initiator, we have $PK_M = PK_A$, and $\mathcal{M}$ resides between $\mathcal{A}$ and $\mathcal{B}$. A scenario for the UKS attack is depicted in Figure 3. When $\mathcal{A}$ sends $\{X, KP\{x\}\}$ to $\mathcal{B}$, $\mathcal{M}$ interrupts the communication and establishes a session with $\mathcal{B}$. $\mathcal{M}$ simply sends $\{X, KP\{x\}\}$ to $\mathcal{B}$. $\mathcal{B}$ verifies $KP\{x\}$, and sends $\{Y, KP\{y\}\}$ back to $\mathcal{M}$ that will be forwarded to $\mathcal{A}$. $\mathcal{A}$ verifies $KP\{y\}$, and believes that she is communicating with $\mathcal{B}$, while $\mathcal{B}$ believes that he is communicating with $\mathcal{M}$. Both $\mathcal{A}$ and $\mathcal{B}$ will reach to the same session key $k = H(g^{(x+a)(y+b)})$.

A simple solution that does not increase the computational cost and guarantees invulnerability of the YAK protocol to the UKS attack is to modify the session key derivation function as $k = H(g^{(x+a)(y+b)}), \hat{A}, \hat{B})$. The other solution is to enforce ZKPs to include unique identifiers with bindings to the corresponding digital certificates. This requires a NIZK scheme employed in the protocol. It can make the protocol vulnerable to the UKS attack if one uses a ZKP scheme without binding to the identifiers.

4.3 Key-replication attack

In a key-replication attack [1], an attacker establishes two different non-matching sessions that output the same session key, in which one of the two sessions (the test session) is unexposed. If such an attack is successful, the adversary can easily find the session key of the test session by querying the second non-matching session which contains the same session key. This is allowed since the sessions are non-matching.

The YAK protocol is vulnerable to the key-replication attack which invalidates the semantic security of the protocol in any security model that such an attack is feasible. A scenario for the key-replication attack on the YAK protocol is depicted in Figure 3. Although the attack scenario in Figure 3 seems similar to the UKS attack, but they have different goals and views. The UKS attack is typically a practical attack wherein an entity has an incorrect belief about its peer in the key exchange. The key-replication attack is typically a theoretical attack which aims to win the security game and invalidate the semantic security of a protocol in the corresponding security models. For a key-replication attack on the YAK protocol, an adversary $\mathcal{M}$ just needs to start two different sessions with $\mathcal{A}$ and $\mathcal{B}$ simultaneously. $\mathcal{M}$ just forwards messages that she receives from one party to the other party. The established session key of these two sessions will be the same. As two established sessions are non-matching, $\mathcal{M}$ can issue a *Reveal* query to the second session which contains the same session key, and get the corresponding session key in response. As the session keys of two non-matching sessions are the same, $\mathcal{M}$ could indeed obtain the session key of the test session. This invalidates the semantic security of the protocol in all security models that a successful key-replication attack does not violate the freshness condition. A countermeasure to the key-replication attack is to include the session identifiers in the key derivation function [32]. The session identifiers can be defined as transcripts of the protocol run. Then, two non-matching sessions will not yield the same session key, and the key-replication attack will not be feasible.

4.4 Further defects

4.4.1 Impersonation attack

The YAK protocol is claimed to be an AKE protocol. However, the authentication in the YAK protocol is just ZKP verification of a random number, generated by the other party. No secret information has been used in authentications. The protocol is two-pass, and does not include any implicit key confirmation. The YAK protocol is then susceptible to an impersonation attack if it is two-pass as designed, and is not followed by extra steps for the key confirmation. An adversary can impersonate $\mathcal{A}$ or $\mathcal{B}$. Here is how the attack works: The adversary generates a random number, and generates proofs for his knowledge of the random number. As the authentication at the other party is just knowledge proofs of the random number, the adversary will be verified and authenticated. Of course, the adversary cannot generate a session key, but there is no further step for the key confirmation. There are practical scenarios where this can be useful.

Another scenario is through an extended key compromise impersonation (extended-KCI) attack [33] that will be successful even though the YAK protocol is followed by further steps for an explicit key confirmation. In an extended-KCI attack, an adversary has access to both long-term and ephemeral private keys of a victim [33]. This attack is not allowed in many security models including the CK01 and eCK models, and is disputed in [34] as it can trivially break many protocols.

4.4.2 Small subgroup attack

If the YAK protocol is followed by extra steps for the key confirmation which is crucial to thwart an impersonation attack mentioned in Section 4.4.1, it is crucial that $\mathcal{A}$ and $\mathcal{B}$ check that the ephemeral public key received from the other party is of prime order q. Otherwise, the protocol would be vulnerable to a small subgroup attack, whereby an insider attacker can obtain the static private key of another honest entity [2, 35].

The YAK protocol is a variant of the HMQV protocol if one sets $d = e = 1$ in the HMQV protocol, and adds ZKPs of ephemeral private keys. Then, the small-subgroup attack to the HMQV protocol, mentioned in [2, 35] is applicable to the YAK protocol. For avoiding the small-subgroup attack, both static and ephemeral public keys must be checked to be of order q.

As mentioned in Section 3, it is assumed in [13, 14] that the Schnorr scheme includes extra steps for verifying that $X^q = Y^q = 1$, although there is not such verification in the Schnorr scheme [20]. The author has implicitly assumed that checking $X^q = Y^q = 1$ should be part of any ZKP scheme with the soundness property, although this is not discussed in [13, 14]. It is a shortcoming in the protocol design to postpone and delegate some vital properties to some primitives without clearly specifying the properties that those primitives must provide. Specifically, when the Schnorr's original scheme [20] does not discuss the NIZK variant, and does not include any condition for checking $X^q = Y^q = 1$.

4.4.3 Perfect forward secrecy

Forward secrecy is an important security attribute in AKE protocols. If static private keys of one or more entities are compromised, it should not affect the security of session keys established before the compromise [36]. Based on allowed combinations for compromise of static private keys and the type of attack as either an active or a passive attack, we have the notions of PFS and weak-PFS (wPFS). The notion of wPFS was introduced in the HMQV model [1] where Krawczyk introduced an active attack which can defeat the PFS for a generic two-message key exchange protocol authenticated via public keys and with no secure shared state perviously established between the entities. The wPFS allows an adversary to reveal static private keys of the involved entities, but the adversary is only allowed to accomplish a passive attack. The Krawczyk's generic PFS attack [1] leaded to a belief in the security community that no two-pass AKE protocol can achieve the PFS, and the best they can achieve is the wPFS [8]. Such belief has been disputed in [37] by introduction of $\text{eCK}^{\text{Passive}}$, eCK^{w} and eCK-PFS security models, where a security-strengthening protocol transformation (or compiler) SIG can transform a generic two-message DH type protocol from $\text{eCK}^{\text{Passive}}$ or eCK^{w} models to a two-message protocol which is secure in the eCK-PFS model. The approach includes relaxing the notion of the matching session, and introducing the concept of the *origin-session* [37].

The YAK protocol is claimed to provide "full" forward secrecy as the author argues that the "perfect" forward secrecy makes no sense and has no concrete meaning. By the "full" forward secrecy, the author refers to the situation where static private keys of both involved entities are compromised. However, it is essentially the wPFS. It is trivial to show that the YAK protocol does not provide the PFS as the Krawczyk's generic PFS attack is also feasible on the YAK protocol. The YAK protocol is then insecure in the eCK-PFS security model. Here is the modified Krawczyk's PFS attack on the YAK protocol:

- $\mathcal{M}$ selects $x \in \mathbb{Z}_q$, computes $X = g^x$ and $KP\{x\}$, and sends $\{X, KP\{x\}\}$ to $\mathcal{B}$.
- $\mathcal{B}$ verifies $KP\{x\}$, selects $y \in \mathbb{Z}_q$, computes $Y = g^y$ and $KP\{y\}$, and sends $\{Y, KP\{y\}\}$ (supposedly to $\mathcal{A}$). $\mathcal{B}$ completes the session by computing the session key as $k = H((X.PK_A)^{y+b})$.
- Once the session key expires at $\mathcal{B}$, and is removed from the memory, $\mathcal{M}$ corrupts $\mathcal{A}$ and finds the private key a. This is allowed because the matching session was not established at $\mathcal{A}$ who did not choose the value x. $\mathcal{M}$ can compute the session key as $k = H((Y.PK_B)^{x+a})$ which contradicts the PFS property.

A simple solution to the PFS problem is adding an explicit key confirmation [1] which increases number of passes. Another solution is to deploy the SIG compiler [37] which does not increase number of passes in the modified protocol, but requires each entity to have two independent pair of public/private keys. The extra key pair is required for the digital signature scheme in the SIG compiler.

5 Conclusion

The security of the YAK protocol [13, 14] has been analyzed in this paper. The YAK protocol is based on public keys, certified by a certificate authority, and then requires a PKI. It is a variant of the two-pass HMQV protocol where the XCR signatures are removed, but zero-knowledge proofs of ephemeral private keys are added to the protocol. In this paper, we show that the YAK protocol lacks the joint key control and PFS attributes, and is vulnerable to several attacks including the unknown key-share and key-replication attacks. This invalidates the semantic security of the protocol in many security models including the HMQV and eCK security models. There are also other considerations regarding the impersonation and small subgroup attacks that were discussed in this paper. Although the YAK protocol is claimed to be an authenticated protocol, the authentication is just zero-knowledge proof of a random number, generated by the other party. A key confirmation is then a must to avoid an impersonation attack. In this case, it is necessary to check the prime order of ephemeral public keys in order to avoid the small subgroup attacks. We also proposed some solutions for thwarting the mentioned attacks on the YAK protocol.

Acknowledgement

The author would like to thank Øyvind Ytrehus and all the anonymous reviewers for their helpful comments.

References

[1] Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In *Advances in Cryptology–CRYPTO'05*, vol. 3621 of *Lecture Notes in Computer Science*, pp. 546–566. Springer Berlin Heidelberg, 2005.

[2] Menezes, A.: Another look at HMQV. *Mathematical Cryptology JMC* 1(1), 47–64, 2007.

[3] Toorani, M.: On continuous after-the-fact leakage-resilient key exchange. In *Proceedings of the 2nd Workshop on Cryptography and*

Security in Computing Systems (CS2'15), pp. 31–34. ACM, January 2015.

[4] Toorani, M.: On vulnerabilities of the security association in the IEEE 802.15.6 standard. In *Financial Cryptography and Data Security*, LNCS 8976, pp. 245–260. Springer Berlin Heidelberg, 2015.

[5] Toorani, M.: Cryptanalysis of a protocol from FC'10. In *Financial Cryptography and Data Security*, LNCS 8975, p. 569. Springer Berlin Heidelberg, 2015.

[6] Bellare, M., Rogaway, P.: Entity authentication and key distribution. In *Advances in Cryptology–CRYPTO'93*, LNCS 773, pp. 232–249. Springer Berlin Heidelberg, 1994.

[7] Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology – EUROCRYPT'01*, LNCS 2045, pp. 453–474. Springer Berlin Heidelberg, 2001.

[8] LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In *Provable Security*, LNCS 4784, pp. 1–16. Springer Berlin Heidelberg, 2007.

[9] Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. *Design Codes Cryptogr.* 28(2), 119–134, 2003.

[10] Toorani, M.: Cryptanalysis of a new protocol of wide use for email with perfect forward secrecy. *Security and Communication Networks* 8(4), 694–701, 2015.

[11] Toorani, M.: Security analysis of J-PAKE. In *Proceedings of the 19th IEEE Symposium on Computers and Communications (ISCC'14)*, pp. 1–6. 2014.

[12] Mitchell, C., Ward, M., Wilson, P.: Key control in key agreement protocols. *Electronics Letters* 34(10), 980–981, May 1998.

[13] HAO, F.: On robust key agreement based on public key authentication. *Security and Communication Networks* 7(1), 77–87, 2014.

[14] HAO, F.: On robust key agreement based on public key authentication. In *Financial Cryptography and Data Security*, LNCS 6052, pp. 383–390. Springer Berlin Heidelberg, 2010.

[15] BELLARE, M., ROGAWAY, P.: Provably secure session key distribution: The three party case. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC'95, pp. 57–66. ACM, New York, NY, USA, 1995.

[16] BELLARE, M., POINTCHEVAL, D., ROGAWAY, P.: Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology–EUROCRYPT'00*, LNCS 1807, pp. 139–155. Springer Berlin Heidelberg, 2000.

[17] TOORANI, M.: Cryptanalysis of two PAKE protocols for body area networks and smart environments. *International Journal of Network Security* 17(5), 629–636, 2015.

[18] CREMERS, C. J.: Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS authenticated key exchange protocol. In *Applied Cryptography and Network Security*, LNCS 5536, pp. 20–33. Springer Berlin Heidelberg, 2009.

[19] USTAOGLU, B.: Comparing SessionStateReveal and EphemeralKeyReveal for Diffie-Hellman protocols. In *Provable Security*, vol. 5848 of *Lecture Notes in Computer Science*, pp. 183–197. Springer Berlin Heidelberg, 2009.

[20] SCHNORR, C.: Efficient signature generation by smart cards. *Journal of Cryptology* 4(3), 161–174, 1991.

[21] LIM, C. H., LEE, P. J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Advances in Cryptology–CRYPTO'97*, LNCS 1294, pp. 249–263. Springer Berlin Heidelberg, 1997.

[22] Toorani, M., Beheshti, A.: A directly public verifiable signcryption scheme based on elliptic curves. In *Proceedings of the 14th IEEE Symposium on Computers and Communications (ISCC'09)*, pp. 713–716. 2009.

[23] Toorani, M.: SMEmail - a new protocol for the secure e-mail in mobile environments. In *Proceedings of the Australian Telecommunications Networks and Applications Conference (ATNAC'08)*, pp. 39–44. IEEE, 2008.

[24] Toorani, M., Beheshti, A.: SSMS - a secure SMS messaging protocol for the m-payment systems. In *Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC'08)*, pp. 700–705. July 2008.

[25] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO'86*, LNCS 263, pp. 186–194. Springer Berlin Heidelberg, 1987.

[26] Sarr, A., Elbaz-Vincent, P., Bajard, J.-C.: A secure and efficient authenticated Diffie-Hellman protocol. In *Public Key Infrastructures, Services and Applications*, LNCS 6391, pp. 83–98. Springer Berlin Heidelberg, 2010.

[27] Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography* 46(3), 329–342, 2008.

[28] Toorani, M., Beheshti, A.: Cryptanalysis of an efficient signcryption scheme with forward secrecy based on elliptic curve. In *Proceedings of International Conference on Computer and Electrical Engineering (ICCEE'08)*, pp. 428–432. Dec 2008.

[29] Kaliski, B. S.: An unknown key-share attack on the MQV key agreement protocol. *ACM Transactions on Information and System Security (TISSEC)* 4(3), 275–288, 2001.

[30] Toorani, M., Beheshti, A.: Cryptanalysis of an elliptic curve-based signcryption scheme. *International Journal of Network Security* 10(1), 51–56, 2010.

[31] Boyd, C., Cremers, C., Feltz, M., Paterson, K. G., Poettering, B., Stebila, D.: ASICS: Authenticated key exchange security incorporating certification systems. In *Computer Security - ESORICS 2013*, vol. 8134 of *Lecture Notes in Computer Science*, pp. 381–399. Springer Berlin Heidelberg, 2013.

[32] Choo, K.-K., Boyd, C., Hitchcock, Y.: On session key construction in provably-secure key establishment protocols. In *Progress in Cryptology – Mycrypt 2005*, LNCS 3715, pp. 116–131. Springer Berlin Heidelberg, 2005.

[33] Tang, Q., Chen, L.: Extended KCI attack against two-party key establishment protocols. *Information Processing Letters* 111(15), 744 – 747, 2011.

[34] Basin, D., Cremers, C., Horvat, M.: Actor key compromise: Consequences and countermeasures. In *IEEE 27th Computer Security Foundations Symposium (CSF'14)*, pp. 244–258. July 2014.

[35] Menezes, A., Ustaoglu, B.: On the importance of public-key validation in the MQV and HMQV key agreement protocols. In *Progress in Cryptology–INDOCRYPT'06*, LNCS 4329, pp. 133–147. Springer Berlin Heidelberg, 2006.

[36] Toorani, M., Beheshti, A.: An elliptic curve-based signcryption scheme with forward secrecy. *Journal of Applied Sciences* 9(6), 1025–1035, 2009.

[37] Cremers, C., Feltz, M.: Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal. *Designs, Codes and Cryptography* 74(1), 183–218, 2015.