Towards Trustworthy Cryptographic Protection of Airplane Information Assets

Basel Alomair, Krishna Sampigethaya, Andrew Clark, and Radha Poovendran
Network Security Lab (NSL), EE Dept., University of Washington, Seattle, WA, 98195, USA

Digital signature is emerging as an accepted solution for protecting aircraft assets during their storage and distribution over computer networks in aviation information systems (AIS). However, the design of trustworthy signature protocols that can address the unique challenges of AIS is an open problem. This paper presents candidate signature protocols that are able to address some of the major challenges, such as recovering from unanticipated loss of secrecy of signing keys and accommodating delegation of signing authority at ground entities interacting with aircraft. These protocols are shown to be computationally secure. Specifically, the paper proposes a generic construct for making standard signature schemes to be forward secure, ensuring that any exposure of the signing key does not lead to forgery and repudiation of signatures produced. Further, an extension of this construct is given for building forware secure proxy signature schemes that enable multiple authorized entities to assume the role of asset signer on behalf of, for example, an airline.

I. Introduction

Modern aircraft, also known as the e-enabled aircraft, will be highly integrated with large-scale distributed systems on the ground for exchanging information.[1,2] The resulting spectrum of aviation information systems (AIS) will handle information assets critical to safe and dependable operation of aircraft such as loadable software, electronic flight bag, onboard configuration reports and maintenance data.[3,4]

However, use of cheap off-the-shelf solutions and open data networks in the AIS present vulnerabilities for unauthorized access to and manipulation of assets. Attacks on an aircraft's critical assets can give rise to safety concerns, e.g., tampering loadable software assessed at the Radio Technical Commission for Aeronautics (RTCA) DO-178B safety levels A-D[7] can potentially degrade airplane airworthiness as well as present airline business threats, e.g., engineering a late detection of asset corruption or false alarm detection can create unwarranted flight delays and costs.[4,14,15]

Digital signatures offers a solution approach for protecting the integrity and authenticity of aircraft assets.[4] A signed asset from a source (*signer*) to a destination (*verifier*) in the AIS can be informally described as:

$$asset, sign_{signer}(H(asset), timestamp), cert_{signer}.$$

$sign_x(.)$ denotes signature of an entity x and $H(.)$ is a one-way cryptographic hash. In order to verify the signature, a valid certificate of the signer is needed:

$$cert_{signer} = sign_{CA}(signer, K_{signer}, CA, validity\_period).$$

The CA is the Certificate Authority, a trusted third party which is an integral part of the public key infrastructure which we assume is available in the AIS to support signatures. The signer certificate can be validated using the CA's valid public key and checking the validity period. Therefore, assuming the CA's public key is known, the verifier can use the contents of $cert_{signer}$ and the *timestamp* to verify the *integrity* and *authenticity* of the received asset.

Current literature well addresses the security and implementation challenges to the use of digital signatures in the AIS, however, they do not cover the design of digital signature schemes that can be trusted.[1,4–6]

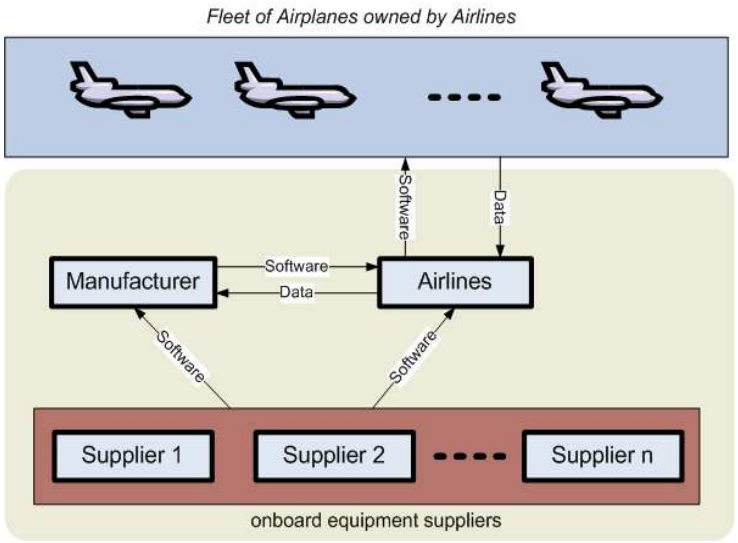

Figure 1. Illustration of the aviation information system (AIS) distributing loadable software and other critical assets of airplanes such as configuration reports and onboard health diagnostics.

Therefore, this paper focuses on trustworthy schemes for signing aircraft assets. Specifically, the paper studies the design of signature schemes that address major AIS vulnerabilities and constraints discussed in the next section.

The rest of the paper is as follows. Section II describes the system model, the major vulnerabilities and constraints considered. Section III overviews the candidate solution approaches that address these major AIS vulnerabilities and constraints. Section IV presents the proposed forward secure signature scheme. Section V describes the proposed forward secure proxy signature scheme. Finally, Section VI concludes the paper.

II. System Model

Fig. 1 shows a generic AIS for distributing critical airplane assets. Apart from distribution of assets between onboard and airline ground systems, assets are also communicated between airlines, airframe manufacturer, and/or from the onboard equipment suppliers. The use of digital signatures can secure assets either hop-to-hop, e.g., between airlines and airplane in Fig. 1, or end-to-end, e.g., supplier to airplane in Fig. 1. We assume that the adversary intends to create safety concerns and business disruptions by attempting to steal secret quantities. Such attacks result in the following major threat to the AIS.

Key Exposure. Key exposure is one of the biggest threats to the security of standard signatures in the AIS, resulting from the exposure of the cryptographic signing key. With some signing keys residing on aircraft line replaceable units and several airlines being relatively new to key management processes, key exposure presents a potential vulnerability. An adversary with access to an exposed key can forge signatures that are indistinguishable from the signatures of authorized entity, potentially inserting corrupted software into the aircraft systems without detection to disrupt airworthiness or airline business. Furthermore, all the signatures of authorized entity become repudiable, even if they have been generated much before the key exposure, creating liability concerns in the event of hazards.

Additionally, we assume that the signing task at each entity in the AIS may be performed by many, resulting in the following major constraint.

Delegation of Signing. The distributed nature of the AIS makes the delegation of the signing authority necessary at some entities. For example, with multiple personnel and ground systems involved in signing assets delivered from an airlines to its fleet, the airlines should be able to delegate its signing authority to multiple entities.

III. Overview of Proposed Solutions

A. Forward Secure Signatures (FSS)

In order to minimize the damage caused by key exposure, the paper considers *forward-secure signature schemes (FSS)*, a concept put forth by Anderson[8] and formalized by Bellare and Miner.[9] In FSS, although an adversary with access to exposed keys can generate valid signatures, the validity of signatures generated prior to the key exposure will remain intact. Consequently, forged signatures with past dates are distinguishable from valid signatures. In a FSS scheme, time is divided into disjoint intervals, say T periods t_1, t_2, ..., t_T ; each period t_i has a secret key, while the public key remains the same. At the end of each interval, a new secret key is generated and the secret key corresponding to the previous interval is deleted.[9] Hence, FSS is time dependent, i.e., a signature must be correlated in some way to the time when the signature is generated. On the other hand, a verifier in the FSS must also have a mechanism to verify that the signature generated during interval t_i is uniquely related to the secret key that is valid at t_i. To ensure forward-secrecy, however, it is required that old secret keys cannot be computed by unauthorized users based on the knowledge of present or future keys.

An assured approach to design FSS is to apply a generic construction to standard signature schemes.[8,9] Generic schemes have advantages such as flexibility to be instantiated from different standard signature schemes and can be provably secure assuming secure standard signature schemes exist.[12] However, a major challenge to the design of FSS schemes is resolving the validity of signatures generated within the key exposure interval.[9] Obviously, all signatures generated before the key exposure but within the same interval will be repudiable, since the same key is used throughout the entire interval. Therefore, the design of interval lengths can be a nontrivial task. The longer the interval, the more the signatures generated with the same key, hence violating the whole idea behind forward-security in digital signatures. On the other hand, shorter intervals will result in a more frequent key updates, even if no signature has been generated during the intervals.

As will be described in Section IV, the paper proposes a generic construction to compose a FSS scheme for AIS. This construct can be applied to any standard signature scheme based on the Discrete Logarithm Problem (DLP). Unlike existing FSS designs which correspond keys with time intervals, the proposed approach ties keys with signatures, i.e., each key is used for one and only one signature. After every signature generation the key is updated independent of time.

B. Forward-Secure Proxy Signatures (FSPS)

In order to enable delegation of signing authority in the AIS, the paper considers *forward-secure proxy signatures (FSPS)*. Unlike standard signatures, in the proxy signature setup, introduced by Mambo et. al.,[13] there are two legitimate users – the proxy designator Alice and the proxy signer Bob. Each one of them is assumed to possess a pair of registered private and public keys, respectively. Alice can delegate her signing power to Bob. An advantage of the above generic FSS construct is that it is extensible to proxy signature schemes, as will be shown in Section V.

IV. A Generic Forward Secure Signature Construction

Throughout the rest of the paper we assume the existence of public key infrastructure, at which users possess registered private and public key pair (x, y), where x represents the private key and y represents the public key. Depending on context, the term signing key is used interchangeably with the terms private or secret key. Similarly, the terms public and verifying key are used synonymously. Since we will construct a FSS from any standard signature scheme based on the discrete logarithm problem (DLP), we start by defining a DLP-based standard signature scheme.

Definition 1 (Standard Digital Signature Scheme)**.** *A standard digital signature scheme* $\mathcal{SS}= (\mathcal{P}, \mathcal{K}, \mathcal{S}, \mathcal{V})$*, with* $\mathcal{P}, \mathcal{K}, \mathcal{S}$*, and* $\mathcal{V}$ *being polynomial-time algorithms with the following functionalities.*

1. $\mathcal{P}$ *is a randomized parameter-generating algorithm that, on input* 1^k*, where* k *is a security parameter, outputs a description of a multiplicative group* $\mathcal{G}$*, a generator* g*, and a description of a one-way hash*

function. These parameters are assumed to be publicly known.

2. *$\mathcal{K}$ is a randomized key-generating algorithm that takes the output of $\mathcal{P}$ as input and outputs a pair of keys (x, y), where x is a secret key and y is the corresponding public key.*

3. *$\mathcal{S}$ is a possibly randomized signing algorithm that takes as input a message $M \in \{0,1\}^*$ and a secret key x. The algorithm outputs a signature σ on the message M.*

4. *$\mathcal{V}$ is a deterministic verification algorithm that takes as input (M, y, σ), such that:*

$$\mathcal{V}(M, y, \sigma) = \begin{cases} 1, & \text{if } \sigma = \mathcal{S}(M, x) \\ 0, & \text{otherwise} \end{cases} . \tag{1}$$

Equation (1) demands that the verification algorithm $\mathcal{V}$ outputs 1 only if the signature σ on message M is generated using the secret key x corresponding to the public key y. Otherwise put, the verification algorithm $\mathcal{V}$ outputs 1 only if the signature is valid.

A. The Proposed Construct

The proposed construction is a modified version of Alomair *et al.*;[11] forward-security is achieved using a *forward-security chain*, R. The forward-security chain is generated off-line and is not required for signature generation nor is it required to be kept secret. To describe our construction method, let $\mathsf{SS}=(\mathcal{P}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be a standard digital signature scheme as in Definition 1. Based on SS, the constructed forward-secure signature scheme is $\mathsf{FSS}=(\mathcal{P}, \mathcal{K}, \mathcal{FKG}, \mathcal{FS}, \mathcal{FV}, \mathcal{KU})$, where $\mathcal{P}, \mathcal{K}, \mathcal{FKG}, \mathcal{FS}, \mathcal{FV}$, and $\mathcal{KU}$ are polynomial-time algorithms. The algorithms $\mathcal{P}$ and $\mathcal{K}$ are exactly the same as in the base scheme. The forward-secure key generation algorithm $\mathcal{FKG}$, the forward-secure signing algorithm $\mathcal{FS}$, the forward-secure verifying algorithm $\mathcal{FV}$, and the forward-secure key update algorithm $\mathcal{KU}$ are described in detail below.

KEY GENERATION. On input of a security parameter l, the user generates a prime p and a prime q that divides $p-1$, such that $q \geq 2^l$. The user picks an element $g \in \mathbb{Z}_p^*$ of order q, and selects a hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The parameters p, q, g, and h are assumed to be publicly known.[a] With the above public parameters and the total number of periods for the forward-secure scheme T in hand, the signer generates a forward-security chain $R = (r_1, r_2, ..., r_T)$, where each r_i corresponds to i^{th} time interval.

Algorithm 1 $\mathcal{FKG}(T)$

$k_1 \xleftarrow{R} \mathbb{Z}_q^*$;
$r_1 \leftarrow g^{-k_1} \pmod p$;
for $i = 2, ..., T$ **do**
  $k_i \leftarrow h(k_{i-1})$;
  $r_i \leftarrow g^{-k_i} \pmod p$;
  `Delete` k_{i-1}
end for
$R \leftarrow (r_1, r_2, ..., r_T)$;
`Return` R

To start, the signer generates an integer k_1 picked randomly from the multiplicative group $\mathbb{Z}_q^*$. The value of r_1 is then computed from k_1 as:

$$r_1 = g^{-k_1} \pmod p. \tag{2}$$

Using the one-way hash function h, the signer continues to construct a chain of k_i's:

$$k_i = h(k_{i-1}), \tag{3}$$

[a] This setup is the same as in the Schnorr signature scheme.[10] For different standard signature schemes, setup varies according to the used standard scheme.

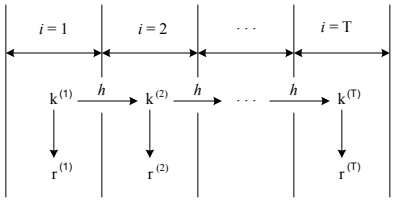

Figure 2. The forward-security chain generation: Secret key for a given signature is a hash of the secret key for the previous signature.

of length T. For each k_i the corresponding r_i is computed as in equation (2). Figure 2 illustrates an implementation of the key generation phase.

The function h in equation (3) must be a one-way function so that evaluating k_{i-1} from k_i can be assumed infeasible. Moreover, by the discrete logarithm assumption, computing k_i using the knowledge of r_i is infeasible.

After the forward-security chain R has been generated, the signer uses her registered secret key, x, to sign the chain (using any secure standard signature scheme). The secret key x is only needed to sign the chain R in the key generation phase and should not be stored in the system. Otherwise, an adversary breaking into the system can forge a signature for any R. Note that the only parameter that the signer is required to store after the completion of the key generating phase is the value of k_1. The chain R is used to provide forward-security and is not required for signature generation. Observe that the key generation is performed only once during the lifetime of the FSS, and it is performed off-line.

Signature Generation. To sign a message during the i^{th} period, the signer uses the corresponding k_i to run the signing algorithm $\mathcal{FS}$. That is, the signer calls the base signing algorithm with k_i as the signing key.

$$s = \mathcal{S}(M, k_i), \tag{4}$$

where $\mathcal{S}$ represents the signing algorithm corresponding to the standard signature scheme used as a building block. The tuple $\sigma = (i, s, r_i)$ comprises the signature on message M. Further improvement can be made, depending on the resources available for the signer. If computational efficiency is more important than storage, the signer can store R in the system. Storing R in the system will save the signer one modular exponentiation by passing r_i as a parameter (instead of recalculating it inside the signing oracle).

Algorithm 2 $\mathcal{FS}(M, i, k_i)$

$s \leftarrow \mathcal{S}(M, k_i)$
`Return` $\sigma = (i, s, r_i)$

Signature Verification. The verification algorithm $\mathcal{FV}$ is shown below. The verifier uses r_i to verify the validity of the signature, using the standard verification algorithm $\mathcal{V}$. Then, the verifier runs the standard signature verification again to verify the validity of the forward-security chain R, using the signer's public key y, and verifies that the i^{th} element of R is equal to r_i. Note that, at this stage, the verifier must also get R and its signature from the sender, if he or she has not already done so.

Algorithm 3 $\mathcal{FV}(M, \sigma, R, y, \sigma_R)$

if $\mathcal{V}(M, r_i, \sigma) = 1$ **then**
  if $\mathcal{V}(R, y_1, \sigma_R) = 1$ `and` $r_{i_\sigma} = r_{i_R}$ **then**
    `Return` 1
  else
    `Return` 0
  end if
else
  `Return` 0
end if

Observe that, without the second check, which is to check the validity of the chain R and if the value r_i in the signature σ is equal to the authenticated r_i in R, the verification algorithm is just the verification algorithm of the base scheme. That is, the scheme can be used as a standard scheme and, if needed (e.g., in case of a dispute), R can be used to ensure forward-security. Furthermore, note that R only needs to be verified once by the receiver.

KEY UPDATE. After the $i - 1^{st}$ period has been elapsed, the signer updates the secret key k_{i-1} by applying the one-way hash function to get k_i. As soon as the value of k_i has been computed, the value of k_{i-1} must be deleted to ensure forward-security, as can be seen in algorithm $\mathcal{KU}$ below.

Algorithm 4 $\mathcal{KU}(k_{i-1})$

$k_i \leftarrow h(k_{i-1})$
`Delete` k_{i-1}
`Return` k_i

To illustrate the construction of the presented FSS, Table 1 details the construction of an FSS using the Schnorr signature scheme as a building block.[10]

B. Security Analysis

We propose the following definition of security for our cipher. Similar models have been proposed in the past, for instance in.[9]

Definition 2 (Forward Security)**.** *Let $\mathcal{A}$ be any polynomial-time adversary. This definition is designed to model forward security under the assumption that* all *secret keys are revealed to the adversary at some point. We define the following game between the signer and $\mathcal{A}$.*

1. *The challenger runs $\mathcal{P}$, $\mathcal{K}$, and $\mathcal{FKG}$ and gives the public key to $\mathcal{A}$.*
2. *$\mathcal{A}$ interacts with the following oracles:*
   - ***Sign:** $\mathcal{A}$ can ask for a signature on an arbitrary message M for the current time period.*
   - ***Update:** $\mathcal{A}$ can decide to move forward in time by asking the challenger to increment the time period and update the secret key.*
   - ***Break-in:** $\mathcal{A}$ requests that the challenger give up all secret keys (in our case, we exclude the secret key x, since it is deleted from secure storage after the $\mathcal{FKG}$ phase). After executing a break-in, $\mathcal{A}$ cannot make any more queries of any oracle.*
3. *$\mathcal{A}$ comes up with a message m and a signature σ for some time period $t < t'$, where t' is the time period when the break-in occurred. $\mathcal{A}$ is successful if $\mathcal{FV}(m, \sigma, R, y, \sigma_R) = 1$ and the adversary had not previously queried message m at time period t.*

Note that we do not include x, the key used to sign the security chain R, in our definition of forward security. Since this key should be deleted after key generation, we assume that it could not be recovered by an adversary during key update.

Table 1. Applying the proposed construction method to design an FSS based on the Schnorr signature scheme.[10] The key generation $\mathcal{FKG}$, signature generation $\mathcal{FS}$, signature verification $\mathcal{FV}$, and the key updating $\mathcal{KU}$ algorithms are summarized.

Algorithm $\mathcal{FKG}(T)$
$k_1 \xleftarrow{R} \mathbb{Z}_q^*$
$r_1 \leftarrow g^{-k_1} \pmod p$;
For $i = 2, ..., T$ **do**
$k_i \leftarrow h(k_{i-1})$;
$r_i \leftarrow g^{-k_i} \ (mod\ p)$;
Delete k_{i-1}
EndFor
$R \leftarrow (r_1, r_2, ..., r_T)$;
Return R
	Algorithm $\mathcal{FS}(M, i, k_i)$
	$r_i \leftarrow g^{-k_i} \pmod p$;
	$\gamma \xleftarrow{R} \mathbb{Z}_q^*$;
	$s \leftarrow h(M, i, r_i)k_i + \gamma \pmod q$;
	$\lambda \leftarrow g^{\gamma} \pmod p$;
	Return $\sigma = (i, s, r_i, \lambda)$

Algorithm $\mathcal{FV}(M, \sigma, R, y, \sigma_R)$
If $\lambda \equiv r_i^{h(M,i,r_i)} g^s \pmod p$
If $\mathcal{V}(R, y_1, \sigma_R) = 1$ **and** $r_{i_\sigma} = r_{i_R}$
Return 1
Else
Return 0
EndIf
Else
Return 0
EndIf
	Algorithm $\mathcal{KU}(k_{i-1})$
	$k_i \leftarrow h(k_{i-1})$;
	Delete k_{i-1}
	Return k_i

Theorem 1. *Given the security of the underlying base signature scheme, to break the forward-security of the proposed scheme, the forger must solve the discrete logarithm problem or invert the one-way hash function.*

Proof. Assume a forger has broken into the system during time interval t_i, thus obtaining k_i. To forge a signature on a message, m, that corresponds to a time interval t_{i-j}, for some $j \in \mathbb{N}^+$, the forger must know k_{i-j}. (Because any signature during time interval t_{i-j} must be signed with the secret key corresponding to the authenticated r_{i-j} in the forward-security chain R.) Since k_{i-j} has been deleted from the system, the forger can recover it from k_i only if she can invert the hash function. On the other hand, to recover k_{i-j} from its corresponding r_{i-j}, the forger must be able to solve the discrete logarithm. □

V. Proposed Forward Secure Proxy Signatures

In proxy signature schemes, Alice delegates her signing capability to Bob. The idea of digital proxy signatures was first introduced by Mambo *et al.*[13] Many of the proposed proxy signature schemes appeared in the literature are based on the following concept: Alice has a pair of keys (x_a, y_a), where x_a and y_a represent the secret and public keys, respectively. To delegate her signing power to the Bob, Alice generates a warrant describing Bob's authorities to sign messages on her behalf. The warrant is then signed by Alice (using a standard signature scheme) and sent to Bob. After checking the validity of the signature, Bob combines Alice's signature with his secret key x_b to generate a *proxy* signing key x_p. Bob uses the *proxy* signing key x_p to sign messages on behalf of Alice using a standard signature scheme. To validate a proxy signature, the verifier computes the public key y_p corresponding to the proxy secret key x_p (usually a function of Alice's and Bob's public keys; that is, $y_p = f(y_a, y_b)$) and use the corresponding standard signature verification

Table 2. A forward-secure proxy signature scheme constructed by applying the proposed construction method to the proxy signature scheme in.[16] Assuming the forward-security chain has been generated successfully, the proxy key generation $\mathcal{PKG}$, proxy signature generation $\mathcal{PS}$, proxy signature verification $\mathcal{PV}$, and the proxy key updating $\mathcal{PKU}$ algorithms are summarized. The subscript a indicates parameters generated by the proxy designator and w represents the warrant describing the authority given to the proxy, as in the original scheme in.[16] x_b represents the registered secret key of the proxy signer, while x_p represents the key generated to sign messages.

Algorithm $\mathcal{PKG}(w, \sigma_a, i, k_i)$ $r_i \leftarrow g^{k_i} \pmod p$ $x_{p_i} \leftarrow h(i, r_i)k_i + h(w, r_a)x_a + k_a \pmod q$ **Return** x_{p_i}	**Algorithm** $\mathcal{PS}(M, x_{p_i})$ $s_p \leftarrow h(M)x_{p_i} + \gamma \pmod q$; $\lambda \leftarrow g^{\gamma} \pmod p$; **Return** $\sigma_M = (i, r_i, s_p, \lambda, w, r_a)$;

Algorithm $\mathcal{PV}(y_a, y_b, M, \sigma_M, R, \sigma_R)$
$y_{p_i} \leftarrow r_i^{h(i,r_i)} y_a^{h(w,r_a)} r_a \pmod p$;
If $\lambda \equiv y_{p_i}^{-h(M)} g^{s_p} \pmod p$
If $\mathcal{V}(R, y_b, \sigma_R) = 1$ **and** $r_{i\sigma} = r_{iR}$
Return 1
Else
Return 0
EndIf
Else
Return 0
EndIf
	Algorithm $\mathcal{PKU}(k_{i-1})$
	$k_i \leftarrow h(k_{i-1})$;
	Delete k_{i-1}
	Return k_i

algorithm to verify the signature.

The idea here is the same idea for constructing regular FSS. For lack of space, we omit describing the details of the construction and outline the basic concept. The major difference between the standard proxy signature scheme and the forward-secure version is in the proxy and key initialization stage, which we describe below.

Proxy and key Initialization. The proxy key generation is an interactive protocol. To start the protocol, Alice decides the number of signatures T for the forward-secure proxy scheme. Upon receiving T, Bob runs the same algorithm $\mathcal{FKG}$ used in the construction of non-proxy forward-secure signature schemes to generate the forward-security chain R of length T, signs it with his key x_b, and sends it to Alice. Alice generates a warrant containing R and her delegation agreement, signs it with her private key x_a, and sends it to Bob.

To sign a message M during the i^{th} time interval, the pair (k_i, r_i) is used by Bob to generate forward-secure proxy signatures with the private key x_p.

Table 2 illustrates our construction of a forward-secure proxy signature scheme based on the provable secure scheme of Kim *et al.*[16] The constructed scheme in Table 2 assumes that the proxy and key initialization has been performed successfully and the forward-security chain R is available for verifiers.

The security analysis of the forward-secure proxy scheme is similar to the non-proxy one. Assuming the standard proxy scheme is secure[b], provided that the forward-security chain R has been generated successfully,

[b]Kim *et al.*[16] is an example of a provable secure proxy signature scheme.

the forward-security is granted by the hardness of the discrete logarithm problem and the existence of one-way functions.

VI. Conclusions and Future Work

In this paper, we proposed trustworthy signature protocols that can protect aircraft information assets, despite exposure of signing keys and the presence of multiple entities responsible for signing assets. The generic construction based forward secure signature protocols can use any standard signature as the underlying scheme. We then showed how the proposed generic construction method can be easily extended to any proxy signature scheme to obtain forward secure proxy signatures.

A major challenge presented by the AIS is the service lifetime of an aircraft, which is in the order of several decades. This imposes the need for long-term signatures for airplane assets. In order to extend the lifetime of signatures, potential approaches include use of a large key length and more robust signature algorithms. Alternatively, periodic signature refresh offers an approach that may be applicable, but this requires careful consideration of the efficiency and impact of periodically updating the signatures of an aircraft's assets. Such long-term signature solution approaches and tradeoffs associated with their use in the AIS will be considered in our future work.

References

[1]C. Wargo and C. Dhas, "Security considerations for the e-enabled aircraft," *Proceedings of Aerospace Conference*, 2003.

[2]Sampigethaya, K., Poovendran, R., Bushnell, L. Secure operation, maintenance and control of future e-enabled airplanes, *Proceedings of the IEEE*, Vol. 96, No. 12, Dec. 2008, pp. 1992-2007.

[3]G. Bird, M. Christensen, D. Lutz, and P. Scandura, "Use of integrated vehicle health management in the field of commercial aviation," in *Proceedings of NASA ISHEM Forum*, 2005.

[4]R. Robinson, M. Li, S. Lintelman, K. Sampigethaya, R. Poovendran, D. von Oheimb, J. Busser, and J. Cuellar, "Electronic distribution of airplane software and the impact of information security on airplane safety," in *Proceedings of the International Conference on Computer Safety, Reliability and Security (SAFECOMP)*, 2007.

[5]R. Robinson, K. Sampigethaya, M. Li, S. Lintelman, R. Poovendran, and D. von Oheimb, "Challenges for it infrastructure supporting secure network-enabled commercial airplane operations," in *Proceedings of AIAA Infotech@Aerospace Conference*, 2007.

[6]R. Robinson, M. Li, S. Lintelman, K. Sampigethaya, R. Poovendran, D. von Oheimb, and J. Busser, "Impact of public key enabled applications on the operation and maintenance of commercial airplanes," in *Proceedings of the AIAA Aviation Technology, Integration and Operations (ATIO) Conference*, 2007.

[7]RTCA, Software Considerations in Airborne Systems and Equipment Certification (RTCA/DO-178B), 1992.

[8]R. Anderson, "Two remarks on public key cryptology," invited lecture, *Proceedings of ACM CCS*, 1997.

[9]M. Bellare and S. Miner, "A forward-secure digital signature scheme," *Proceedings of CRYPTO*, pp. 431448, 1999.

[10]C. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, pp. 161–174, 1991

[11]B. Alomair, K. Sampigethaya, and R. Poovendran, "Efficient Generic Forward-Secure Signatures and Proxy Signatures," *European Public Key Infrastructure*, pp. 166–181, 2008

[12]E. Cronin, S. Jamin, T. Malkin, P. McDaniel, "On the performance, feasibility, and use of forward-secure signatures," *Proceedings of 10th ACM conference on Computer and communications security*, pp. 131144, 2003.

[13]M. Mambo, K. Usuda, E. Okamoto, "Proxy Signatures: Delegation of the Power to Sign Messages," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 79(9), pp. 13381354, 1996.

[14]Federal Aviation Administration, 14 CFR Part 25, Special Conditions: Boeing model 7878 airplane; systems and data networks securityisolation or protection from unauthorized passenger domain systems access, [Docket No. NM364 Special Conditions No. 250701SC], Federal Register, Vol. 72, No. 71., 2007, http://edocket.access.gpo.gov/2007/pdf/E7-7065.pdf

[15]Federal Aviation Administration, 2007, 14 CFR Part 25, Special Conditions: Boeing model 7878 airplane; systems and data networks securityprotection of airplane systems and data networks from unauthorized external access, [Docket No. NM365 Special Conditions No. 250702SC], Federal Register, Vol. 72, No. 72., 2007, http://edocket.access.gpo.gov/2007/pdf/07-1838.pdf

[16]S. Kim, S. Park, and D. Won, "Proxy signatures, Revisited," *Proceedings of the First International Conference on Information and Communication Security*, pp. 223–232, 1997.