An Efficient Real-time Broadcast Authentication Scheme for Command and Control Messages

Attila A. Yavuz, *Member, IEEE*

***Abstract*—Broadcast (multicast) authentication is crucial for large and distributed systems such as cyber-physical infrastructures (e.g., power-grid/smart-grid) and wireless networks (e.g., inter-vehicle networks, military ad-hoc networks). These time-critical systems require real-time authentication of command and control messages in a highly efficient, secure and scalable manner. However, existing solutions are either computationally costly (e.g., asymmetric cryptography) or unscalable/impractical (e.g., symmetric cryptography, one-time signatures, delayed key disclosure methods).**

In this paper, we develop a new broadcast authentication scheme that we call *Rapid Authentication (RA)*, which is suitable for *time-critical* authentication of command and control messages in *large and distributed* systems. We exploit the semi-structured nature of command and control messages to construct special digital signatures, which are computationally efficient both at the signer and verifier sides. We show that *RA* achieves several desirable properties that are not available in existing alternatives simultaneously: (i) Fast signature generation and verification; (ii) immediate verification; (iii) constant size public key; (iv) compact authenticating tag; (v) packet loss tolerance; (vi) being free from time synchronization requirement; (vii) provable security.

***Index Terms*—Secure Broadcast Authentication; Applied Cryptography; Security of Networked Systems; Network Security.**

I. Introduction

A broadcast (multicast) authentication scheme enables each receiver in a large broadcast group to verify if the received message is intact and originated from the claimed sender. Broadcast authentication is a vital security service for many real-life applications (e.g.,wireless networks [1]–[3]).

Secure broadcast authentication is a challenging problem, especially for large and distributed systems with time-critical applications [4]. For instance, cyber-physical infrastructure such as power-grid/smart-grid requires real-time authentication of control messages in an efficient, scalable and secure manner [5]. Other similar examples include vehicle-to-vehicle/infra-structure communication [6], disaster response systems (e.g., fire-sensors, earthquake warning) and military networks. In all these systems, immediate and secure authentication of command and control messages is essential to prevent adversaries from forcing catastrophic decisions by modifying or forging the messages [7]–[9].

Attila A. Yavuz is with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331 USA, (e-mail: `attila.yavuz@oregonstate.edu`). Work done in part while at Robert Bosch LLC, Research and Technology Center at North America (CR/RTC3-NA), Pittsburgh, PA.

A. Related Work

We present an overview of existing broadcast authentication techniques and discuss their advantages and limitations.

- *Traditional Symmetric and Asymmetric Cryptography Methods*: Symmetric cryptography based authentication methods rely on Message Authentication Code (MAC) [10] to achieve computational efficiency. Despite their simplicity, these methods are not practical to be used for broadcast (multicast) authentication purposes in large and distributed systems [4], [11]. That is, they require a pairwise key distribution between the signer(s) and verifiers to be secure, which makes them impractical even for moderately large systems. Moreover, due to their symmetric nature, these schemes are not publicly verifiable and therefore cannot achieve the non-repudiation property.

Traditional Public Key Cryptography (PKC)-based schemes (e.g., digital signatures) [10] rely on public key infrastructures. Hence, they are publicly verifiable, key compromise resilient and scalable for large systems. However, these schemes (e.g., RSA [12], ECDSA [13]) require Expensive Operations (ExpOps)[1] such as modular exponentiation. These high computational costs make PKC-based schemes impractical for real-time (e.g., power-grid/smart-grid [7]) and resource-constrained applications.

- *Delayed Key Disclosure Methods*: The objective of delayed disclosure methods (e.g., TESLA [3] and its variants [14], [15]) is to achieve public verifiability of symmetric primitives while retaining their computational efficiency. Intuitively, a MAC is appended to every message and the key of the MAC is disclosed in some subsequent packet. Disclosed keys are computed based on hash chains to increase the packet loss tolerance. Despite their advantages, these methods have the following drawbacks:

(i) Receivers cannot verify a message until its corresponding keying material is received (i.e., immediate verification is not possible). Such a delay is not tolerable for time-critical applications (e.g., power-grid/smart-grid). (ii) These schemes require a tight time synchronization between the sender and all receivers. Maintaining a continuous synchronization is challenging for large and distributed systems.

- *Signature Amortization Methods*: These methods (e.g., [16]–[18]) compute a digital signature over a set of messages instead of computing a distinct signature for each message. Hence, the cost of signature generation and verification is amortized over multiple messages.

[1]For brevity, we denote an expensive cryptographic operation such as modular exponentiation, elliptic curve scalar multiplication or cryptographic pairing as an ExpOp.

However, signature amortization methods do not allow verification of an individual message in a given window until all related messages are received. Therefore, they cannot achieve the immediate verification. These schemes are also vulnerable to packet loss [19], since there is a correlation between different messages. Moreover, they still require ExpOps to compute and verify the signatures.

• *One-Time Signatures (OTSs)*: OTSs achieve high computational efficiency and public verifiability since they rely on one-way functions without trapdoors. Preliminary OTSs [20] require very large private/public key and signature sizes. Later, more efficient constructions have been proposed (e.g., [21], [22]). One of the most efficient OTSs is the Hash-to-Obtain Random Subset (HORS) [22], which offers a fast signature generation/verification and relatively smaller signature size than its precessors (e.g., BiBa [21]). Despite its elegancy, HORS has the following limitations:

(i) A private/public key pair can be used only once without losing the security. It is also possible to compute a (small) limited number of signatures with the same private/public key by sacrificing the security (e.g., [21], [22]) and performance (e.g., [23], [24]). Therefore, all OTSs (including HORS) require distribution of new public keys, which causes heavy communication overhead. (ii) The public key/signature size of HORS is large and the certification (or chaining) of public keys incurs additional overhead. (iii) Various twists on HORS have been proposed to achieve different performance trade-offs (e.g., lower storage with the expense of very high computational cost [8]) or security trade-offs (e.g., TV-HORS [7]). However, these schemes inherit the drawbacks of HORS. Moreover, they drastically decrease the efficiency of one parameter while only slightly increasing the efficiency of an another parameter.

• *Online/offline Signatures*: The online/offline signatures shift expensive signature computations to the offline phase (these operations can be performed without knowing the actual message to be signed). This allows a fast signature generation in the online phase. Traditional online/offline signatures use OTSs as a building block. Hence, they inherit all the drawbacks of OTSs such as large public key and signature size. Recently, space efficient online/offline signatures have been proposed (e.g., [25]). However, they incur high computational overhead at the verifier side due to ExpOps.

B. Our Contributions

The above discussions indicate that a broadcast authentication scheme achieving minimum end-to-end cryptographic delay with a constant signature and key sizes is needed.

One may obverse that a signer efficient and compact RSA-based signature scheme is a suitable solution. That is, given that RSA is already verifier efficient (e.g., for public key $e = 3$), such a signer efficient RSA-based signature achieves a low end-to-end cryptographic delay. However, as discussed in Section I-A, existing methods such as RSA (or Rabin [29]) based online/offline signatures incur large signature and key sizes. Similarly, RSA does not offer a natural way to pre-compute tokens that could permit fast signature generation (in contrast to some DLP-based alternatives such as ECDSA [13] that allow pre-computed tokens [26]).

In this paper, we develop a new signature scheme called *Rapid Authentication (RA)*, which exploits already existing structures in command and control messages to enable pre-computation for signature schemes like RSA. Examples of such semi-structured command and control messages can be found in tactical networks (e.g., military or disaster recover networks [30]) and proprietary communication protocols (e.g., local energy distribution systems).

The main idea of RA is to leverage the fact that the number of possible sub-messages in a command and control message are limited. Hence, it is possible to pre-compute and store a RSA signature on each of those sub-messages during the offline phase. When a message to be signed in the online phase, the signer combines individual RSA signatures of relevant sub-messages via *Condensed-RSA* [31], which requires only a few modular multiplications. The verification of this signature is also efficient, as it requires a standard RSA signature verification plus a few modular multiplications. Moreover, we introduce a signature masking technique to protect individual signatures combined via Condensed-RSA. We show that RA is secure (in Random Oracle Model (ROM) [28]) and is also more efficient than its counterparts.

C. Desirable Properties

We summarize the desirable properties of RA and compare it with the existing alternatives below:

- *High Signer and Verifier Computational Efficiency*: RA does not require any ExpOp in its online phase. That is, the total end-to-end computational overhead of RA is just a small number of multiplications and hash operations.
  - The computational efficiency of RA is at least *an order of magnitude* greater than traditional PKC-based signature schemes (e.g., [12], [13], [29], [32]).
  - RA is also more computationally efficient and much more scalable than HORS variants (e.g., [7], [8], [33]) without requiring a security and performance trade-off.
- *High Scalability and Communication Efficiency*: OTS schemes (e.g., HORS [22] and other alternatives such as [7], [8], [21], [22], [33]) require pre-distribution and retransmission of large size public keys. This also brings related problems such as certification or chaining of such public keys. Similar to the traditional PKC-based signatres, RA uses a single public key to verify (practically) unbounded number of signatures, which makes it much more communication efficient. This communication efficiency also makes RA more scalable than those alternatives.
- *Compact Public Key and Signature*: The public key and signature size of our scheme is nearly the same with its building block scheme RSA [12]. Hence, it is much more compact than OTS schemes, which have large public key and signature sizes (e.g., [7], [20]–[22]).
- *Immediate Verification and being Free from the Time Synchronization Requirement*: RA does not rely on delayed key disclosure or message buffering. Therefore, unlike TESLA

TABLE I
HIGH-LEVEL COMPARISON OF *RA* AND THE STATE-OF-ART SCHEMES WITH RESPECT TO VARIOUS PROPERTIES

	RA	PKC		OTS		*Online/Offline [25]*	*TESLA*	*Symmetric* (pairwise)
		RSA/ECDSA	Amortized	HORS	TV-HORS			
1. Practical for Time-Critical Applications	Yes	No	No	Yes	Yes	No	No	Yes
2. Scalability for Large and Distributed Systems	High	High	High	Low	Moderate	High	High	Very Low
3. Free from Public Key Re-Distribution Problem	Yes	Yes	Yes	No	Partial	Yes	Yes	Yes
4. Free from Synchronization Need	Yes	Yes	No	Yes	No	Yes	No	Yes
5. Immediate Ver. (no buffering/delayed disclosure)	Yes	Yes	No	Yes	Yes	Yes	No	Yes
6. Free from Time-Bounded Security	Yes	Yes	Yes	Yes	No	Yes	No	Yes
7. Non-repudiation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
8. Packet Loss Tolerance	Full	Full	Partial	Full	Partial	Full	Partial	Full
9. Real-Time (end-to-end) Computational Efficiency	High	Very Low	Low	High	Moderate	Very Low	High	High
10. Communication Efficiency (tag size)	High	High	High	Low	High	Low	High	Low
11. Verifier Storage Efficiency	High	High	High	Low	Moderate	High	High	High
12. Signer Storage Efficiency	Low	Varies	Varies	Low	Moderate	Moderate	High	Moderate

Quantitative properties (row 9-12) such as computation, storage and communication efficiency are evaluated according to the analytical/experimental results given in Section V. Qualitative properties (row 1-8) are justified based on the discussions provided in Section I-A. For instance, PKC-based schemes are not practical for time-critical applications due to their high (end-to-end) computational overhead. Symmetric schemes have linear tag size (with the pairwise setting) that makes them communication inefficient and unscalable for large and distributed systems. The signer storage efficiency of ECDSA (and its amortized version) varies depending on its implementation. For example, tokenized ECDSA [26] eliminates ExpOps from the signer side with the expense of a linear storage overhead. HORS is unscalable due to its public key re-distribution requirement and storage/communication inefficiency. TV-HORS offers a security-performance trade-off and therefore it has a moderate/partial performance for certain properties. We provide a more detailed discussion in Section V.

TABLE II
NOTATION USED IN OUR PAPER

$\|\|, \|x\|, \{0,1\}^*$: Concatenation operation, bit lengths of variable x, the set of binary strings of any finite length, respectively.
$\|\mathcal{S}\|$ denotes the cardinality of set $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ denotes that variable x is randomly and uniformly selected from set $\mathcal{S}$.
Z_n^* denotes a multiplicative group, where n is a large integer.
For any integer l, $(x_0, \ldots, x_l) \xleftarrow{\$} \mathcal{S}$ means $(x_0 \xleftarrow{\$} \mathcal{S}, \ldots, x_l \xleftarrow{\$} \mathcal{S})$.
Full Domain Hash (FDH) (e.g., [27]) function H is defined as $H : \{0,1\}^* \rightarrow Z_n^*$.

This table summarizes notations used in our scheme. Large integer n is a RSA public key as defined in Definition 1. In our scheme, FDH H is modeled as an ideal hash function (i.e., a Random Oracle as defined in [28]).

variants (e.g., [3], [15]) and signature amortization techniques (e.g., [19]), it can address real-time applications.

- *Individual Message Authentication and High Robustness*: In RA, receivers can verify each individual message received. Hence, RA is much more resilient against the packet loss than signature amortization techniques (e.g., [16], [19]).

Table I summarizes the high-level comparison of RA with its counterparts. A detailed performance analysis and comparison is given in Section V.

Limitations: RA requires that the broadcast entity is resourceful in order to store pre-computed message-signature tables. This requirement is reasonable for our envisioned applications, in which the signers are generally resourceful entities such laptops, command centers or base stations.

Note that RA leverages the semi-structured nature of command and control messages to achieve its desirable properties. Hence, RA is *not* suitable for the applications, in which the message content is randomized and non-structured.

The remainder of this paper is organized as follows. Section II gives the definitions and RA system and security models. Section III describes RA in detail. Section IV gives a detailed security analysis of RA. Section V presents performance analysis and compares RA with previous approaches. Section VI concludes this paper.

II. DEFINITIONS AND MODELS

In this section, we first give definitions used by RA. We then provide the system and threat and security models of RA, respectively.

We summarize basic notations used in our paper in Table II.

A. Definitions

RA is based on RSA [12] and condensed-RSA [31], which are defined below:

Definition 1 RSA signature scheme is a tuple of three algorithms (Kg, Sig, Ver) defined as follows:

- $(sk, PK) \leftarrow RSA.Kg(1^\kappa)$: The key generation algorithm takes the security parameter 1^κ as the input. It randomly generates two large primes (p, q) and computes $n = p \cdot q$. The public and secret exponents $(e, d) \in Z_n^*$ satisfies $e \cdot d \equiv 1 \bmod \phi(n)$, where $\phi(n) = (p-1)(q-1)$. It returns a private/public key pair $sk \leftarrow d$ and $PK \leftarrow (n, e)$ as the output.
- $\sigma \leftarrow RSA.Sig(sk, m)$: The signature generation algorithm takes sk and a message $m \in \{0,1\}^*$ as the input. It returns a signature $\sigma \leftarrow [H(m)]^d \bmod n$ as the output (also denoted as $\sigma \leftarrow RSA.Sig_{sk}(m)$).

- $c \leftarrow RSA.Ver(PK, m, \sigma)$: The signature verification algorithm takes PK, m and σ as the input (also denoted as $c \leftarrow RSA.Ver_{PK}\ (m, \sigma)$). If $\sigma^e = H(m) \bmod n$ it returns bit $c = 1$ meaning *valid*. Otherwise, it returns $c = 0$ meaning *invalid* .

Definition 2 Condensed-RSA scheme (denoted as $C\text{-}RSA$) is a tuple of three algorithms (Kg, Sig, Ver) defined as follows:
- $(sk, PK) \leftarrow C\text{-}RSA.Kg(1^\kappa)$: Execute $RSA.Kg(1^\kappa)$ in Definition 1.
- $\sigma \leftarrow C\text{-}RSA.Sig(sk, \overrightarrow{m})$: The signature generation algorithm takes sk and a set of messages $\overrightarrow{m} = (m_0, \ldots, m_l)$ as the input. It returns a signature $\sigma \leftarrow \prod_{j=0}^{l} \sigma'_j \bmod n$ as the output, where $\sigma'_j \leftarrow [H(m_j)]^d \bmod n$ for $j = 0, \ldots, l$ (also denoted as $\sigma \leftarrow C\text{-}RSA.Sig_{sk}(\overrightarrow{m})$).
- $c \leftarrow C\text{-}RSA.Ver(PK, \overrightarrow{m}, \sigma)$: The signature verification algorithm takes PK, $\overrightarrow{m}$ and σ as the input (also denoted as $c \leftarrow C\text{-}RSA.Ver_{PK}(\overrightarrow{m}, \sigma)$). If $\sigma^e = \prod_{j=0}^{l} H(m_j) \bmod n$ it returns bit $c = 1$ meaning *valid*. Otherwise, it returns $c = 0$ meaning *invalid* .

We give the cryptographic interfaces of RA below, which are used in our security model (i.e., Definition 4). The detailed description of RA algorithms is given in Section III.

Definition 3 RA is a tuple of four algorithms, $(Kg, \mathit{Offline\text{-}Sig}, \mathit{Online\text{-}Sig}, Ver)$ defined as follows:
- $(sk, PK) \leftarrow RA.Kg(1^\kappa)$: The key generation algorithm takes the security parameter 1^κ as the input. It returns a private/public key pair (sk, PK) as the output.
- $\overline{sk} \leftarrow RA.\mathit{Offline\text{-}Sig}(sk, \overrightarrow{M})$: The offline signature generation algorithm is executed offline before the system deployment. It takes the private key sk and a vector of pre-defined message sets $\overrightarrow{M} = (M_0, \ldots, M_l)$ as the input, where each M_j is a set for $j = 0, \ldots, l$. It returns a cryptographic token $\overline{sk} = (\Gamma, \overrightarrow{\beta})$ as the output, where Γ and $\overrightarrow{\beta}$ denote pre-computed signature and message sets (defined in Section III).
- $\sigma \leftarrow RA.\mathit{Online\text{-}Sig}(\overline{sk}, \overrightarrow{m})$: The online signature generation algorithm takes the cryptographic token $\overline{sk}$ (i.e., the private key of the online phase) and an online message $\overrightarrow{m} = (m_0, \ldots, m_l)$ as the input. It returns a signature $\sigma = (r, s)$ as the output, which is comprised of a random number r and a condensed-RSA signature s.
- $c \leftarrow RA.Ver(PK, \overrightarrow{m}, \sigma)$: The signature verification algorithm takes PK, a message set $\overrightarrow{m}$ to be verified and its corresponding signature σ as the input. It outputs a bit c, with $c = 1$ meaning *valid* and $c = 0$ meaning *invalid*.

B. System Model

Our system model relies on the PKC-based broadcast authentication model (e.g., [21], [22]), which includes two types of entities:

(i) A resourceful broadcast entity (i.e., the signer), which broadcasts message-signature pairs to the receivers. The signer is assumed to be trusted and it cannot be compromised by the adversary. This is compatible with our envisioned applications, in which the signer is storage capable and trusted such as a command center, base station or a satellite. (ii) Computational, storage and bandwidth limited receivers (e.g., verifiers). A verifier can be any (untrusted) entity (e.g., a sensor) and it might be compromised by adversary.

We assume that the signer executes the key generation and initial offline signature generation before the deployment. After the deployment, the signer can execute the offline signature generation either on-demand or periodically. Note that the offline phase must be performed independently from the online phase to achieve a high real-time computational efficiency.

C. Threat and Security Model

Our threat model reflects how a PKC-based broadcast authentication scheme works. That is, adversary $\mathcal{A}$ can observe message-signature pairs computed under sk. $\mathcal{A}$ also can actively intercept, modify, inject and replay messages transmitted over the network.

A standard security notion that captures our threat model is Existential Unforgeability under Chosen Message Attack ($EU\text{-}CMA$) [34].

RA is proven to be a $EU\text{-}CMA$ signature scheme based on the experiment defined in Definition 4. In this experiment, $\mathcal{A}$ is provided with a signing oracle $RA.\mathit{Online\text{-}Sig}_{sk}(.)$. $\mathcal{A}$ can adaptively query $RA.\mathit{Online\text{-}Sig}_{sk}(.)$ on any message $\overrightarrow{m}$ she wants up to L queries in total. The signing oracle returns the corresponding RA signature of $\overrightarrow{m}$ under sk. Finally, $\mathcal{A}$ outputs a forgery $(\overrightarrow{m}^*, \sigma^*)$ under PK. If this forgery is valid and non-trivial (i.e., $\mathcal{A}$ did not query σ^* before), $\mathcal{A}$ wins the $EU\text{-}CMA$ experiment. Otherwise, $\mathcal{A}$ loses in the $EU\text{-}CMA$ experiment.

Definition 4 $EU\text{-}CMA$ experiment for RA is as follows:
$(sk, PK) \leftarrow RA.Kg(1^\kappa)$,
$(\overrightarrow{m}^*, \sigma^*) \leftarrow \mathcal{A}^{RA.\mathit{Online\text{-}Sig}_{sk}(\cdot)}(PK)$,
If $RA.Ver(PK, \overrightarrow{m}^*, \sigma^*) = 1$ and $\overrightarrow{m}^*$ was not queried to the signing oracle, return 1, else it return 0.

$\mathcal{A}$'s advantage is $Adv_{RA}^{EU\text{-}CMA}(\mathcal{A}) = Pr[Expt_{RA}^{EU\text{-}CMA}(\mathcal{A}) = 1]$. $Adv_{RA}^{EU\text{-}CMA}(t, L, \mu) = \max_{\mathcal{A}}\{Adv_{RA}^{EU\text{-}CMA}(\mathcal{A})\}$ denotes the probability advantage of RA, where the maximum is over all $\mathcal{A}$ having time complexity t, making at most L oracle queries, and the sum of lengths of these queries being at most μ bits.

In Section IV, we prove that RA is $EU\text{-}CMA$ based on the fact that $C\text{-}RSA$ is $EU\text{-}CMA$, which was proven in [31]. $EU\text{-}CMA$ experiment for $C\text{-}RSA$ is given below.

Definition 5 $EU\text{-}CMA$ experiment for $C\text{-}RSA$ is as follows:
$(sk, PK) \leftarrow C\text{-}RSA(1^\kappa)$,
$(\overrightarrow{m}^*, \sigma^*) \leftarrow \mathcal{A}^{C\text{-}RSA.Sig_{sk}(\cdot)}(PK)$,
If $C\text{-}RSA.Ver(PK, \overrightarrow{m}^*, \sigma^*) = 1$ and $\overrightarrow{m}^*$ was not queried to the signing oracle, return 1, else it return 0.
$\mathcal{A}$'s advantage is $Adv_{C\text{-}RSA}^{EU\text{-}CMA}(\mathcal{A}) = Pr[Expt_{C\text{-}RSA}^{EU\text{-}CMA}(\mathcal{A}) = 1]$. $Adv_{C\text{-}RSA}^{EU\text{-}CMA}(t, L, \mu) = \max_{\mathcal{A}}\{Adv_{C\text{-}RSA}^{EU\text{-}CMA}(\mathcal{A})\}$ denotes the probability advantage of $C\text{-}RSA$, where the maximum is over all $\mathcal{A}$ having time complexity t, making at most L oracle queries, and the sum of lengths of these queries being at most μ bits.

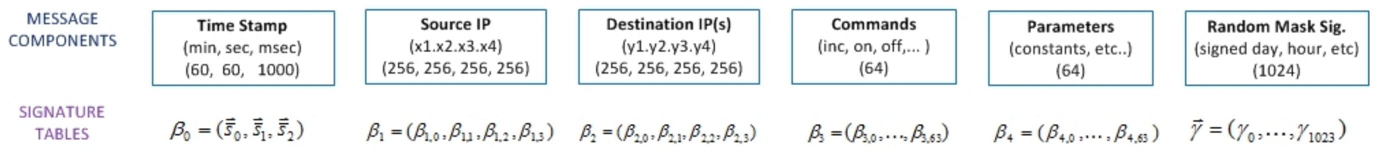
MESSAGE COMPONENTS
Time Stamp
(min, sec, msec)
(60, 60, 1000)
Source IP
(x1.x2.x3.x4)
(256, 256, 256, 256)
Destination IP(s)
(y1.y2.y3.y4)
(256, 256, 256, 256)
Commands
(inc, on, off,...)
(64)
Parameters
(constants, etc..)
(64)
Random Mask Sig.
(signed day, hour, etc)
(1024)
SIGNATURE TABLES
$\beta_0 = (\vec{\vec{s}}_0, \vec{\vec{s}}_1, \vec{\vec{s}}_2)$
$\beta_1 = (\beta_{1,0}, \beta_{1,1}, \beta_{1,2}, \beta_{1,3})$
$\beta_2 = (\beta_{2,0}, \beta_{2,1}, \beta_{2,2}, \beta_{2,3})$
$\beta_3 = (\beta_{3,0}, \ldots, \beta_{3,63})$
$\beta_4 = (\beta_{4,0}, \ldots, \beta_{4,63})$
$\vec{\gamma} = (\gamma_0, \ldots, \gamma_{1023})$

Fig. 1. A representative example of semi-structured messages and its corresponding pre-computed signature table for RA.

III. Description of Our Scheme

In this section, we present our proposed scheme RA. We first give an overview of RA and then provide its detailed description.

A. Overview

We summarize the intuition behind of our scheme below:

a) Semi-Structured Nature of Command and Control Messages: We observe that the content of a command and control message is generally structured in many real-life applications. That is, such a command and control message is semantically fragmented into subsections.

For instance, consider a local energy distribution system (e.g., a smart-town grid system, or a mobile tactical network for disaster recovery), where the control center adjusts the energy parameters of peripheral devices by broadcasting (pre-defined) control messages. A possible structure of such a message is comprised of sender and receiver IP addresses, a command (e.g., reduce voltage, increase frequency), some optional parameters (e.g., constants) and a timestamp with randomizing values to enhance the security (see Figure 1).

b) Pre-computation of Verifier Efficient Signatures (Offline Phase): RA exploits already existing structures in command and control messages to enable pre-computation for a verifier efficient scheme such as RSA [12]. In certain cases, the number of possible sub-messages in a given command and control message (i.e., commands and receiver groups) are limited. Hence, it is practical to pre-compute and store a RSA signature on each of those sub-message components. RA pre-computes sub-message/RSA-signature tables (i.e., $\overrightarrow{\beta}$) during its offline phase (i.e., step 2-b of $RA.Offline\text{-}Sig$ in Section III-B), which eliminates ExpOps from the online phase.

c) Efficient and Secure Combination (Online Phase): Assume that the signer needs to multicast a message $\overrightarrow{m}$={on time t; a fixed signer IP (hence just a single pre-computed condensed-RSA signature of it); receiver IP addresses $y1.y2.y3.y4$ (e.g., a group of receiver); with j-th command; k-th parameter} during the online phase (as in Figure 1). Firstly, the signer fetches the corresponding pre-computed signatures $(sig_t, sig_{SourceIP}, sig_{DestIP}, sig_j, sig_k)$ from sub-message/signature table $\overrightarrow{\beta}$. The signer then combines these signatures into a single and compact RSA signature by multiplying them under modulo n, which produces a valid *Condensed-RSA* [35] signature on message $\overrightarrow{m}$. Note that the online phase of RA is very efficient, since it requires only a few modular multiplications. However, this combination strategy requires further improvements as discussed below:

(i) Freshness and Dynamic Timestamping: Each message must be timestamped to prevent replay attacks. The signature of time stamp is generated via an another time value/signature table (similar to the sub-message/signature table). This prevents the signer from computing a costly online signature for the timestamp (i.e., step 2-b of $RA.Offline\text{-}Sig$ in Section III-B). The optimization described in Remark 2 reduces the storage/computation overhead of these operations.

(ii) One-time Masking: Condensed-RSA is multiplicative homomorphic and mutable[2] signature scheme [31]. Thus, if an adversary $\mathcal{A}$ observes sufficient number of message/signature pairs $\{m_j,\ s_j\}_{j=0}^{l}$, she can recover the individual signatures stored in table $\overrightarrow{\beta}$ by solving multiplicative modular equations revealed via $(s_0, \ldots, s_l)$. $\mathcal{A}$ then can create a valid condensed-RSA signature on any message. There are immutable signature techniques (e.g., [35]), which aim to prevent such an attack. However, these techniques are either interactive or computationally costly (e.g., [35]), which make them impractical.

We address this problem by developing a one-time signature masking technique, which does not require any ExpOp in the online phase. That is, the signer pre-computes a set of RSA signatures γ_j on random numbers $r_j||\overline{r}$. The signer stores these values in a table as $\Gamma = (r_j, \gamma_j)$ for $j = 0, \ldots, l'$, where l' is the total number of random numbers (i.e., step 2-c of $RA.Offline\text{-}Sig$ in Section III-B). In the online phase, the signer randomly picks a (r, γ) pair from Γ and masks the condensed RSA signature of $\overrightarrow{m}$ by multiplying it with γ as $\sigma \leftarrow \gamma \cdot (s)$ (i.e., Equation 1 in $RA.Online\text{-}Sig$ in Section III-B). Since γ is a random number in Z_n^*, this operation *one-time masks* the individual signature components of $\overrightarrow{\beta}$. This solves the aforementioned problem.

d) Scalable and Efficient Verification: The signature verification of RA is equivalent to a condensed-RSA signature verification of the signature component s on message $\widetilde{m} = (r, \overrightarrow{m})$. Therefore, RA signature verification is as efficient as a condensed-RSA signature verification, which is very *fast* for properly selected parameters (e.g., $e = 3$). Similarly, RA uses a single permanent public key PK to verify signatures. Therefore, it is much more *scalable* and *practical* than OTSs (e.g., HORS) requiring public key re-distribution.

Remark 1 One may consider that a system with large number of senders/receivers significantly increases the storage overhead, since the pre-computed signatures of their IP addresses must be stored in $\overrightarrow{\beta}$. We create two sub pre-computed tables for sender and receiver IP addresses. As exemplified in Figure 1, this allows representing all IP addresses with a constant and small set of pre-computed signatures.

[2]Mutability refers that given a set of valid signatures, it is easy to derive new and valid aggregated signatures, which have not been queried before [35].

B. Detailed Description

We give the detailed description of RA below.

1) $(sk, PK) \leftarrow RA.Kg(1^\kappa)$: Generate $\overline{r} \overset{\$}{\leftarrow} \{0,1\}^\kappa$ and a RSA private/public key pair as $(sk', PK') \leftarrow RSA.Kg(1^\kappa)$. Set RA private/public key pair as $sk \leftarrow sk'$ and $PK \leftarrow (PK', \overline{r})$, respectively.

2) $\overline{sk} \leftarrow RA.Offline\text{-}Sig(sk, \overrightarrow{M})$: The offline signature generation algorithm takes a set of message components $\overrightarrow{M}$ and sk as the input. It outputs a signature-message table $\overline{sk} = (\Gamma, \overrightarrow{\beta})$ as follows:

a) *Message Components*: $\overrightarrow{M} \leftarrow \{M_0, \ldots, M_{L-1}\}$ denotes the message components, where L is the total number of message components.
The first component $M_0 = (T_0 || \ldots || T_{k-1})$ denotes the time stamp, where k is the total number of time stamp components in M_0. Each $T_{0 \leq i \leq k-1}$ is also comprised of a set of time values $(t_{i,j}||i||0)$ for $i = 0, \ldots, k-1$ and $j = 0, \ldots, |T_i| - 1$. For instance, given a time format "$yyyy||mm||dd||hh||mm||ss||ms$", we set $k = 7$ (i.e., there are seven time fields in M_0), $|T_1| = 12$ (i.e., the total number of months in T_1) [3].
Each remaining message component $M_{1 \leq i \leq L-1}$ is comprised of a set of messages $m_{i,j}$ for $i = 1, \ldots, L-1$ and $j = 0, \ldots, |M_i| - 1$. For instance, M_1 is the set of commands and M_2 is the set of receivers (e.g., $m_{2,0}$ is the first receiver in M_2).

b) *Compute Message-Signature Tables*: Given M_0, compute a signature on each time value $t_{i,j}$ as $\overline{s}_{i,j} \leftarrow RSA.Sig_{sk}(t_{i,j} ||i||0)$ for $i = 0, \ldots, k-1$ and $j = 0, \ldots, |T_i|-1$. The corresponding signature table of M_0 is $\overline{\beta} = \{\overline{s}_{i,j}\}_{i=0,j=0}^{k-1,|T_i|-1}$. Given $m_{i,j} \in M_i$, compute a signature on $m_{i,j}$ as $s'_{i,j} \leftarrow RSA.Sig_{sk}(m_{i,j}||i)$ for $i = 1, \ldots, L-1$ and $j = 0, \ldots, |M_i| - 1$. The corresponding signature table of M_i is $\beta_i = \{s'_{i,j}\}_{i=1,j=0}^{L-1,|M_i|-1}$.

c) *Compute Random Number-Signature Table*: Compute one-time masking signatures as $r_j \overset{\$}{\leftarrow} \{0,1\}^\kappa$ and $\gamma_j \leftarrow RSA.Sig_{sk}\ (r_j||\overline{r})$ for $j = 0, \ldots, l'$. The corresponding random number/signature pair table is $\Gamma = \{r_j, \gamma_j\}_{j=0}^{l'}$.

d) Assign the private key of online phase $\overline{sk} \leftarrow (\Gamma, \overrightarrow{\beta})$, where $\overrightarrow{\beta} \leftarrow (\beta_0 = \overline{\beta}, \beta_1, \ldots, \beta_{L-1})$.

3) $\sigma \leftarrow RA.Online\text{-}Sig(\overline{sk}, \overrightarrow{m})$: During the online phase, assume that the signer needs to sign a message $\overrightarrow{m} \in \overrightarrow{M}$, where $\overrightarrow{m} = (m_0, \ldots, m_l)$ and m_0 is the current time $(0 < l < L-1)$. Compute the online signature σ as follows:

Fetch the corresponding signatures of time components $m_0 = (t_0|| \ldots ||t_{k-1})$ from β_0 as $(\overline{s}_0, \ldots, \overline{s}_{k-1})$. Fetch the corresponding signatures of remaining message values $(m_1, \ldots, m_l)$ from $\beta_1, \ldots, \beta_l$ as $(s'_1, \ldots, s'_l)$. Last, randomly select a pair (r, γ) from Γ and erase the selected pair from Γ. The signature on $\overrightarrow{m}$ is:

$$s \leftarrow \gamma \cdot (\prod_{j=0}^{k-1} \overline{s}_j \cdot \prod_{i=1}^{l} s'_i) \quad , \quad \sigma \leftarrow (r, s) \tag{1}$$

Individual signatures (i.e., $\overline{s}$ and s') of $\overrightarrow{\beta} = \{\beta_0, \beta_1, \ldots, \beta_{L-1}\}$ are never released. That is, they are protected by being masked by one-time signature γ.

4) $c \leftarrow RA.Ver(PK, \overrightarrow{m}, \sigma)$: Assume that the verifier receives $(\overrightarrow{m}, \sigma)$ on time t' from the signer. The verifier checks whether time stamp m_0 matches with t' and $|r| = \kappa$ hold. If these conditions do not hold, the verifier rejects the signature (i.e., the signature is obsolete or the signature component r is not in the required range). Otherwise, given $m_0 = (t_0|| \ldots ||t_{k-1})$ and $PK = (PK', \overline{r})$, the verifier verifies $\sigma = (r, s)$ as follows:

$$m' \leftarrow H(r||\overline{r}) \cdot (\prod_{j=0}^{k-1} H(t_j||j||0) \cdot \prod_{i=1}^{l} H(m_i||i)) \tag{2}$$

$$c \leftarrow RSA.Ver_{PK'}(m', s) \tag{3}$$

Remark 2 Random number/signature table Γ can be renewed either on-demand or periodically (without disrupting online phase). Periodically renewing Γ enables a lower storage overhead and higher signature generation efficiency. That is, assume that the signer computes a new Γ every day. In this case, all the preceding time components (i.e., years, month) and the given day can be authenticated via the masking signatures as $\gamma \leftarrow RSA.Sig_{sk}\ (r||\overline{r}||t_0||t_1||t_2)$. Hence, the signer stores a smaller number of time stamp signatures (e.g., only for hours, minutes, sec and ms).

Remark 3 $RA.Online\text{-}Sig$ algorithm is equivalent to compute a condensed-RSA signature s on a message $\widetilde{m} = (r, \overrightarrow{m})$. Similarly, one may verify that $RA.Ver$ algorithm is equivalent to verify a condensed-RSA signature s on $\widetilde{m} = (r, \overrightarrow{m})$.

Remark 4 TESLA [3] and its variants (e.g., [15]) rely on time factor to introduce an asymmetric between the signer and verifiers. Hence, they require time synchronization by design. RA, as in traditional signature schemes (e.g., [13]), does not require time synchronization to provide authentication. Notice that if replay attacks are a concern, any digital signature scheme can be coupled with time stamps, which requires a loose time synchronization. However, it is not a requirement of the signature scheme itself, but a requirement of the application (if replay attacks are a concern).

IV. Security Analysis

We prove that RA is an $EU\text{-}CMA$ signature scheme in Theorem 1 below. Note that in our proof, we omit terms that are negligible in terms of κ.

Theorem 1 *$Adv_{RA}^{EU\text{-}CMA}(t, L, \mu)$ is bounded as follows,*

$$Adv_{RA}^{EU\text{-}CMA}(t, L, \mu) \leq Adv_{C\text{-}RSA}^{EU\text{-}CMA}(t', L', \mu'),$$

where $t' = O(t) + L \cdot (RNG + CSIG)$, $\mu' = \mu + L \cdot (2\kappa)$ and $L' = L$.

[3]The time stamp M_0 is the first component of $\overrightarrow{M}$. Therefore, we concatenate 0 to each time value as $(t_{i,j}||i||0)$ to enforce 0'th position of these values in $\overrightarrow{M}$. Index i in $(t_{i,j}||i||0)$ is used to enforce the order of time values within the time stamp M_0.

Proof: Let simulator $\mathcal{A}$ be an RA attacker. We construct a $C\text{-}RSA$ attacker $\mathcal{F}$ that uses $\mathcal{A}$ as a sub-routine. That is, we set $(sk', PK') \leftarrow C\text{-}RSA.Kg(1^\kappa)$ as defined in Definition 5 (i.e., $EU\text{-}CMA$ experiment for $C\text{-}RSA$) and run the simulator $\mathcal{F}$ as follows:

▶ *Algorithm* $\mathcal{F}^{C\text{-}RSA.Sig_{sk'}(\cdot)}(PK')$:

• *Setup:* $\mathcal{F}$ generates $\overline{r} \leftarrow \{0,1\}^\kappa$ and sets the public key for RA as $PK \leftarrow (PK', \overline{r})$ as in $RA.Kg$. Note that $\mathcal{F}$ does not know the private key sk corresponding to PK. However, $\mathcal{F}$ has an access to $C\text{-}RSA$ oracle under $sk' = sk$ and therefore he can simulate RA signatures via this oracle as follows:

• *Execute* $\mathcal{A}^{RA.Online\text{-}Sig_{(.)}(\cdot)}(PK)$: $\mathcal{F}$ replies $\mathcal{A}$'s queries and then check the result of her forgery as below.

- *Queries:* $\mathcal{A}$ adaptively queries data items $\overrightarrow{m}_j = (m_0, \ldots, m_l)$ of her choice to $\mathcal{F}$ for $j = 0, \ldots, L-1$. For each query $\overrightarrow{m}_j$, $\mathcal{F}$ generates a random number $r_j \overset{\$}{\leftarrow} \{0,1\}^\kappa$ and creates a message $\widetilde{m}_j \leftarrow (r_j||\overline{r}, \overrightarrow{m}_j)$. $\mathcal{F}$ then queries $C\text{-}RSA$ oracle on $\widetilde{m}_j$ and obtains the corresponding signature as $s_j \leftarrow C\text{-}RSA_{sk'}(\widetilde{m}_j)$. $\mathcal{F}$ sets the signature on $\overrightarrow{m}_j$ as $\sigma_j \leftarrow (r_j, s_j)$ and returns σ_j to $\mathcal{A}$ as the query answer.
  $\mathcal{F}$ also maintains lists $\mathcal{LD}$ and $\mathcal{LR}$ to keep track the query results during the experiment. For each query j, $\mathcal{F}$ inserts $\overrightarrow{m}_j$ and r_j into $\mathcal{LD}$ and $\mathcal{LR}$, respectively.
- *Forgery of* $\mathcal{A}$: After the query phase, $\mathcal{A}$ outputs a forgery for PK as $(\overrightarrow{m}^*, \langle \sigma^* = (r^*, s^*) \rangle)$. By Definition 4, $\mathcal{A}$ wins if conditions $RA.Ver(PK, \overrightarrow{m}^*, \sigma^*) = 1 \wedge \overrightarrow{m}^* \notin \mathcal{LD}$ hold. If these conditions hold, $\mathcal{A}$ wins the $EU\text{-}CMA$ experiment for RA and returns 1. Otherwise, $\mathcal{A}$ loses in the experiment and returns 0.

▶ *Forgery of* $\mathcal{F}$: If $\mathcal{A}$ loses in the $EU\text{-}CMA$ experiment for RA then $\mathcal{F}$ also loses in the $EU\text{-}CMA$ experiment for $C\text{-}RSA$, and therefore $\mathcal{F}$ aborts and returns 0. Otherwise, $\mathcal{F}$ proceeds as follows:

Given that $\mathcal{A}$'s forgery is $(\overrightarrow{m}^*, \langle \sigma^* = (r^*, s^*) \rangle)$, $\mathcal{F}$ sets $\widetilde{m}^* \leftarrow (r^*, \overrightarrow{m}^*)$ and returns the forgery on PK' as $(\widetilde{m}^*, s^*)$. By Definition 5, $\mathcal{F}$ wins the $EU\text{-}CMA$ experiment for $C\text{-}RSA$ if the following conditions hold: $C\text{-}RSA(PK', \widetilde{m}^*, s^*) = 1 \wedge [(\overrightarrow{m}^* \notin \mathcal{LD}) \vee (r^* \notin \mathcal{LR})]$. If these conditions hold, $\mathcal{F}$ wins the $EU\text{-}CMA$ experiment (i.e., the forgery is valid and non-trivial) and returns 1. Otherwise, $\mathcal{F}$ loses the experiment and returns 0.

The success probability, execution time analysis and indistinguishability argument are as follows:

Success Probability Analysis: The following events are required for $\mathcal{F}$ to succeed in the $EU\text{-}CMA$ experiment:

- $\overline{Abort1}$: $\mathcal{F}$ does not abort due to $\mathcal{A}$'s queries.
  $\mathcal{F}$ simulates $\mathcal{A}$'s queries by expanding each RA query with a 2κ-bit randomness (i.e., $r||\overline{r}$) and requesting the signature of $\widetilde{m} \leftarrow (r||\overline{r}, \overrightarrow{m})$ from $C\text{-}RSA$ oracle. Therefore, $\mathcal{F}$ aborts if and only if he cannot obtain a valid signature from $C\text{-}RSA$ oracle, whose probability is negligible. Therefore, we conclude $Pr[\overline{Abort1}] = 1$.
- $Forge$: $\mathcal{A}$ wins the $EU\text{-}CMA$ experiment for RA.
  If $\mathcal{F}$ does not abort then $\mathcal{A}$ also does not abort, since $\mathcal{A}$'s view in this experiment is perfectly indistinguishable from her view in a real-system (see the indistinguishability argument below). Therefore, the probability that $\mathcal{F}$ does not abort and $\mathcal{A}$ wins the experiment is $Pr[Forge|\overline{Abort1}] = Adv_{RA}^{EU\text{-}CMA}(t, L, \mu)$.
- $\overline{Abort2}$: $\mathcal{F}$ does not abort during his forgery phase.
  The probability that $\mathcal{A}$ wins the experiment without querying $\mathcal{F}$ is negligible as it requires a random guess or finding a collision on H. This guarantees that, by Definition 5 and Remark 3, the forgery $(\widetilde{m}^*, s^*)$ is valid and non-trivial. Therefore, we conclude $Pr[\overline{Abort2}|\overline{Abort1} \wedge Forge] = 1$.
- Win: $\mathcal{F}$ wins the $EU\text{-}CMA$ experiment for $C\text{-}RSA$, whose probability is denoted as $Pr[Win] = Adv_{C\text{-}RSA}^{EU\text{-}CMA}(t', L', \mu')$.

  This occurs if all of the above events happen. That is, $Pr[Win] = Pr[\overline{Abort1}]Pr[Forge|\overline{Abort1}]$ $Pr[\overline{Abort2}|\overline{Abort1} \wedge Forge]$. This equality implies that the $EU\text{-}CMA$ advantage of RA is bounded by the $EU\text{-}CMA$ advantage of $C\text{-}RSA$ as follows:

$$Adv_{RA}^{EU\text{-}CMA}(t, L, \mu) \leq Adv_{C\text{-}RSA}^{EU\text{-}CMA}(t', L', \mu')$$

Execution Time Analysis: The running time of $\mathcal{F}$ is that of $\mathcal{A}$ plus the time it takes to respond L (in total) RA queries. Each RA query requires drawing a random number and requesting a condensed-RSA signature from $C\text{-}RSA$ oracle, whose costs are denoted as RNG and $CSIG$, respectively. Hence, the approximate running time of $\mathcal{F}$ is $t' = O(t) + L \cdot (RNG + CSIG)$.

Note that during the query phase, $\mathcal{F}$ directs each query of $\mathcal{A}$ to $C\text{-}RSA$ oracle by expanding it with a 2κ-bit randomness $r_j||\overline{r}$. Therefore, the total number of queries that $\mathcal{A}$ and $\mathcal{F}$ make are equal as $L' = L$ and the length of $\mathcal{F}$'s query is $\mu' = \mu + L \cdot (2\kappa)$.

Indistinguishability Argument: The real-view of $\mathcal{A}$ is a vector $\overrightarrow{A}_{real} = \{PK, \sigma_j\}_{j=0}^{L}$, where the public key and signatures are computed by $RA.Kg$ and $RA.Online\text{-}Sig$ algorithms. The simulated view of $\mathcal{A}$ is also a vector $\overrightarrow{A}_{sim}$and it is identical to $\overrightarrow{A}_{real}$, where the public key and signatures are computed as follows:

(i) Given $PK = (PK', \overline{r})$, $\mathcal{F}$ directly takes the RSA public key PK' as input and then randomly generates $\overline{r}$ as in $RA.Kg$ algorithm. (ii) During the simulation, $\mathcal{F}$ obtains a condensed-RSA signature s on $(r, \overrightarrow{m})$ from $C\text{-}RSA$ oracle and replies $\mathcal{A}$'s signature query with $\sigma = (r, s)$. The actual $RA.Online\text{-}Sig$ and $RA.Ver$ algorithms are equivalent to generate and to verify a condensed-RSA signature s on $\widetilde{m} = (r, \overrightarrow{m})$ (Remark 3). Hence, the answers of $\mathcal{F}$ for $\mathcal{A}$'s RA queries are valid and perfectly indistinguishable.

The above statements show that all variables in $\overrightarrow{A}_{real}$ and $\overrightarrow{A}_{sim}$ are computed identically. Hence, the joint probability distributions of $\overrightarrow{A}_{real}$ and $\overrightarrow{A}_{sim}$ are equal as $Pr[\overrightarrow{A}_{real} = \overline{a}] = Pr[\overrightarrow{A}_{sim} = \overline{a}]$ (i.e., perfectly indistinguishable). □

V. Performance Analysis

In this section, we present the performance analysis of RA and compare it with previous schemes using the fol-

TABLE III
ANALYTICAL ONLINE (I.E., REAL-TIME) COMPUTATIONAL COST COMPARISON OF *RA* AND PREVIOUS SCHEMES

	RA	Traditional PKC		*Online/Offline [25]*	*(K pre-computed private/public key pairs)*	
		RSA	ECDSA *(K pre-computed tokens)*		HORS	TV-HORS
Signer	$a \cdot Muln$	$Expn$	$H + 2Mul$	$0.1 \cdot Muln$	H	H
Verifier	$(a+e)Muln + a \cdot H$	$e \cdot Muln$	$\approx 1.3 \cdot EMul$	$Expn + Muln$	$u \cdot H$	$\alpha \cdot H$
End-to-end	$(2a+e)Muln + a \cdot H$	$\approx Expn$	$\approx 1.3 \cdot EMul$	$\approx Expn$	$(u+1) \cdot H$	$(\alpha+1) \cdot H$

$Expn$ denotes a modular exponentiation over modulus n. Mul, $Muln$ and $EMul$ denote a modular multiplication over modulus q' (see Table IV-notes), modular multiplication over modulus n and ECC scalar multiplication over q', respectively. e denotes the public key of RSA/condensed-RSA, which is also used as a part of the public key of RA. u, α and a denote the constant parameters used in HORS, TV-HORS and RA, respectively. End-to-end computational overhead is the sum of signer and verifier computational overhead, in which we omit the low-cost operations if there are ExpOps in the sum (e.g., $Expn$). Similarly, small and constant number of additions (e.g., as in ECDSA) are omitted. ECDSA is implemented with tokens [26] at the signer to achieve an ExpOp-free signing. ECDSA signature verification is performed with the double-point scalar multiplication.

TABLE IV
AVERAGE ONLINE EXECUTION TIME (IN μS) COMPARISON OF *RA* AND PREVIOUS SCHEMES (SAMPLED OVER $K = 10^4$ MESSAGES)

	RA*	Traditional PKC		*Online/Offline [25]*	*(K pre-computed private/public key pairs)*	
		RSA	ECDSA *(K pre-computed tokens)*		HORS*	TV-HORS (with a low $\kappa = 54$)
Signer	100	3766	26	2	1	1
Verifier	116	26	1550	1774	20	685
End-to-end	226	3792	1576	1776	21	686

RA^* and HORS* are the most computationally efficient schemes among these alternatives. Note that despite being efficient, HORS is *impractical* as it requires distributing a large public key (e.g., 3KB-5KB) for each message. Table I and Section I-C present a qualitative comparison of RA and HORS. Section V-B presents a storage and communication overhead comparison of RA and HORS.

Parameters: TV-HORS is efficient only with a small κ, since it relies on a security/performance trade-off. Some suggested parameters for TV-HORS [7] are as follows: $\kappa = 54$, packet loss rate $p_l = 0.2$ and $\alpha = 685$. We select standard $\kappa = 80$ for all other schemes with the exception of TV-HORS. Suggested parameters/bit lengths to achieve 80-bit security for other compared schemes are as follows: Composite modulus $|n| = 1024$ for RA, RSA and online/offline construction in [25]. The coefficient $e = 3$ for RA and RSA. Primes $|q'| = 160$ and $|p'| = 512$ for ECDSA. Parameter $u = 20$ for HORS. We select $a = 10$ for RA by following a simplified version of the example given in Figure 1. Sub-fields are as follows: (sec,msec) for timestamp, the fixed signer IP (hence only a single condensed-RSA signature), four sub-fields of receiver IPs, commands, parameters and random masking signatures. The total number of pre-computed signatures for those sub-fields is denoted as c, which is $c = 2213$ (60+1000+1+1024+64+64) for this example.

TABLE V
ANALYTICAL (ASYMPTOTIC) STORAGE AND COMMUNICATION OVERHEAD COMPARISON OF *RA* AND PREVIOUS SCHEMES

		RA	Traditional PKC		*Online/Offline [25]*	*(K pre-computed private/public key pairs)*	
			RSA	ECDSA *(K pre-computed tokens)*		HORS	TV-HORS
Storage (sign/verify	*Signer*	$\|n\|c + (\|n\| + \kappa)O(K)$	$\|n\|$	$(\|p'\| + \|q'\|)O(K)$	$2\|n\|O(K)$	$t\|H\|O(K)$	$t'\|H'\|O(K)/v$
$K = 10^4$ messages)	*Verifier*	$\|n\| + 2\kappa$	$\|n\|$	$\|p'\| + \|q'\|$	$2\|n\|$	$t\|H\|O(K)$	$t'\|H'\|O(K)/v$
Communication (per message)		$\|n\| + \kappa$	$\|n\|$	$\|p'\| + \|q'\|$	$2\|n\|$	$u\|H\|$	$t'\|H'\|$

Suggested parameters/bit length for TV-HORS [7] are ($t' = 12, v = 5, |H'| = 48$) with $\kappa = 54$, where $|H'|$ is the truncated hash output. Suggested parameters for HORS [22] are ($t = 256, u = 20$), where $|H| = 80$. Bit lengths of $|n|, |p'|$ and $|q'|$ are given as in Table IV.

TABLE VI
NUMERICAL STORAGE AND COMMUNICATION OVERHEAD COMPARISON OF *RA* AND PREVIOUS SCHEMES FOR $K = 10^4$ MESSAGES

		RA	Traditional PKC		*Online/Offline [25]*	*(K pre-computed private/public key pairs)*	
			RSA*	ECDSA *(K pre-computed tokens)*		HORS	TV-HORS
Storage (sign/verify	*Signer*	1562 KB	128 byte	101 KB	250 KB	2500 KB	70KB
$K = 10^4$ messages)	*Verifier*	138 byte	128 byte	84 byte	256 byte	2500 KB	70KB
Communication (per message)		138 byte	128 byte	84 byte	256 byte	192 byte	76 byte

After keys for $K = 10^4$ messages depletes, HORS/TV-HORS require distributing K new public keys. Other schemes do *not* require such a public key re-distribution. To achieve the minimum end-to-end delay, all compared schemes (with the exception of plain RSA) are used in pre-computation/pre-distribution setting. That is, all tokens/public keys are pre-computed and distributed before the system deployment. Hence, ECDSA with tokens achieves a higher computational efficiency with the cost of a higher signer storage. A similar principle also applies to HORS and TV-HORS. In RA, $c = 2213$ plus $K = 10^4$ masking signatures create approximately 1526 KB signer storage overhead.

lowing criteria: (i) The computational overhead of signature generation and verification; (ii) storage and communication overhead depending on the size of signature and the size of public key. For each of these criteria, we first analyze RA and then provide a comparison of it with previous schemes. We also discuss some qualitative criteria such scalability and applicability based on the criteria (i)-(ii).

In computational overhead analysis, we focus on *online end-to-end* computational efficiency to evaluate the practicality of compared schemes for real-time applications. In storage and communication overhead analysis, we assume that public keys are generated and distributed before the system deployment.

We select a representative scheme(s) from each main group of schemes discussed in Section I-A. We select RSA [12] and ECDSA [13] as the verifier efficient and signer efficient PKC-based schemes, respectively. We select HORS [22] and TV-HORS [7] as OTS schemes, since HORS is one of the fastest OTSs. We select online/offline signature in [25], which is more scalable than previous online/offline schemes (e.g., [36]).

A. Computational Overhead

Offline (non-real-time) Overhead: The key generation cost of RA is a random number generation plus a single execution of $RSA.Kg$. The $RA.Offline\text{-}Sig$ requires $c \cdot (RSA.Sig)$ computations to prepare message-signature table $\overrightarrow{\beta}$, where c denotes the total number of pre-computed signatures in $\overrightarrow{\beta}$. RA pre-computes a random number-signature table Γ, which requires $K \cdot (RSA.Sig + RNG)$ computations for K messages (Γ is computed as in Remark 2).

Online (real-time) Overhead: The essential computational overhead criteria for all compared schemes is the online (real-time) computational overhead. In RA, the signature generation cost is $a \cdot (Muln)$ for a message with $a = (k + l)$ sub-message components, where k and l are the number of sub-message components for timestamp and for $\overrightarrow{m}$, respectively (as in Equation 1). The signature verification requires a hash computations and a modular multiplications plus a condensed-RSA signature verification (with $e = 3$). That is, the signature verification cost is $(a + e)Muln + a \cdot H$.

Comparison: To achieve the fastest real-time response, for each compared scheme, we shift the ExpOp(s) of signature generation to the offline phase (if the scheme enables this property). That is, we implement an improved version of ECDSA [26], which allows pre-computing and storing cryptographic tokens. These tokens enable ExpOp-free signature generation in the online phase. In HORS and TV-HORS, we generate K one-time private/public key pairs in the offline phase. These private/public keys are stored at the signer and verifier sides, respectively (for immediate use in the online phase). Online/offline signatures (e.g., [25]) by nature realize such a pre-computation at the signer side. Note that, different from its DLP-based counterparts (e.g., ECDSA), such a pre-computation is not known for RSA.

Table III summarizes the analytical comparison of RA with its counterparts. We also prototype all compared schemes on a computer with an Intel(R) Core(TM) i7 Q720 at 1.60GHz CPU and 2GB RAM running Ubuntu 10.10. We measure the execution times using MIRACL [37] library. Table IV provides an average (online) execution time comparison.

Given the example described in Table IV with $(a = 10, e = 3)$, end-to-end delay of RA is as low as 226 μs (without any modular arithmetic optimization). RA is approximately *17, 8, 7 and 3 times faster* than RSA, online/offline scheme [25], ECDSA and TV-HORS (which has a much lower security), respectively. However, RA is less efficient than HORS).

B. Storage and Communication Overhead

In RA, the signer storage overhead is comprised of pre-computed tables $(\overrightarrow{\beta}, \Gamma)$. The size of $(\overrightarrow{\beta}, \Gamma)$ is $|n|c + (|n| + \kappa)O(K)$, where c denotes the total number of pre-computed signatures[4] stored in $\overrightarrow{\beta}$. With optimizations in Remark 2, the signer pre-computes a new Γ from time to time. The size of Γ is $(|n| + \kappa)O(K)$ for K messages to be signed during the designated time interval. Unlike OTS schemes (e.g., HORS [22]), RA does not require distributing public keys once these random numbers deplete. The signer just computes a new Γ in an offline manner. The verifier storage overhead of RA is a constant size public key PK, which is comprised of public key pk' and a κ-bit random number $\overline{r}$.

The communication overhead of RA is $|n| + |\kappa|$ (i.e., the size of signature σ).

Comparison: Table V summarizes the analytical storage/communication overhead comparison of *RA* with its counterparts. Table VI provides numerical values for $K = 10^4$ messages, with parameters in the pre-computation/pre-distribution setting. That is, all keys/tokens [26] are pre-computed and stored for ECDSA, HORS and TV-HORS. RA and online/offline scheme [25] already operate in this setting.

At the verifier side, RA is much more efficient than HORS and TV-HORS and more efficient than online/offline scheme [25]. However, since RA requires storing pre-computed signature tables, it is less efficient than other compared schemes at the signer side, with the exception of HORS.

VI. Conclusion

In this paper, we developed a new broadcast authentication scheme for command and control messages, which we refer to as *Rapid Authentication* (RA). RA simultaneously achieves several desirable properties including fast signature generation and verification, immediate verification without message buffering, small public key and signature size, high scalability, high packet loss tolerance, provable security and being free from synchronization requirement. Our comparison with the existing alternatives shows that RA is an ideal choice for broadcast authentication of command and control messages in large and distributed systems with time-critical applications.

RA achieves its desirable properties by leveraging already existing structures in command and control messages that exist in many real-life applications. However, RA currently cannot provide authentication for command and control messages without pre-defined structures. In future work, we aim designing schemes that can address this limitation of RA.

[4]Sub-message components (e.g., commands and receiver IDs) are already stored as a requirement of the application and therefore their overhead is excluded here.

References

[1] K. Ren, W. Lou, K. Zeng, and P. Moran, "On broadcast authentication in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 11, pp. 4136 –4144, November 2007.

[2] A. Perrig and J. Tygar, *Secure broadcast communication in wired and wireless networks*. Kluwer Academic Publishers, 2003. [Online]. Available: http://books.google.com/books?id=h5qXzbliKNIC

[3] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceedings of the IEEE Symposium on Security and Privacy*, May 2000.

[4] M. Luk, A. Perrig, and B. Whillock, "Seven cardinal properties of sensor network broadcast authentication," in *Proceedings of 4th ACM workshop on security of ad hoc and sensor networks*, ser. SASN '06. New York, NY, USA: ACM, 2006, pp. 147–156.

[5] Y. Liu, M. K. Reiter, and P. Ning, "False data injection attacks against state estimation in electric power grids," in *ACM Conference on Computer and Communications Security*, 2009, pp. 21–32.

[6] H. Guo, Y. Wu, F. Bao, H. Chen, and M. Ma, "UBAPV2G: A unique batch authentication protocol for vehicle-to-grid communications," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 707 –714, December 2011.

[7] Q. Wang, H. Khurana, Y. Huang, and K. Nahrstedt, "Time valid one-time signature for time-critical multicast data authentication," in *INFOCOM 2009, IEEE*, April 2009.

[8] Q. Li and G. Cao, "Multicast authentication in the smart grid with one-time signature," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 686 –696, December 2011.

[9] Z. Lu, X. Lu, W. Wang, and C. Wang, "Review and evaluation of security threats on the communication networks in the smart grid," in *Military Communication Conference (MILCOM)*, November 2010.

[10] A. Menezes, P. C. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996, ISBN: 0-8493-8523-7.

[11] D. Boneh, G. Durfee, and M. K. Franklin, "Lower bounds for multicast message authentication," in *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, ser. EUROCRYPT '01. London, UK, UK: Springer-Verlag, 2001, pp. 437–452. [Online]. Available: http://dl.acm.org/citation.cfm?id=647086.715684

[12] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[13] *ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, American Bankers Association, 1999.

[14] D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical broadcast authentication in sensor networks," in *Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, July 2005, pp. 118 – 129.

[15] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, Sep. 2002.

[16] A. Lysyanskaya, R. Tamassia, and N. Triandopoulos, "Multicast authentication in fully adversarial networks," in *IEEE Symposium on Security and Privacy*, May 2004, pp. 241 –253.

[17] C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts," *IEEE Transaction on Networks*, vol. 7, no. 4, pp. 502–513, August 1999.

[18] S. Miner and J. Staddon, "Graph-based authentication of digital streams," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2001, pp. 232–246.

[19] J. M. Park, E. Chong, and H. Siegel, "Efficient multicast packet authentication using signature amortization," in *IEEE Symposium on Security and Privacy*, May 2002, pp. 227 – 240.

[20] L. Lamport, "Constructing digital signatures from a one-way function," Tech. Rep. CSL-98, October 1979.

[21] A. Perrig, "The BiBa: One-time signature and broadcast authentication protocol," in *Proceedings of the ACM Conference on Computer and Communications Security*, November 2001, pp. 28–37.

[22] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *Proceedings of the 7th Australian Conference on Information Security and Privacy (ACIPS '02)*. Springer-Verlag, 2002, pp. 144–153.

[23] J. Pieprzyk, H. Wang, and C. Xing, "Multiple-time signature schemes against adaptive chosen message attacks," in *Selected Areas in Cryptography (SAC)*, 2003, pp. 88–100.

[24] W. Neumann, "HORSE: An extension of an r-time signature scheme with fast signing and verification," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, vol. 1, april 2004, pp. 129 – 134 Vol.1.

[25] A. Shamir and Y. Tauman, "Improved online/offline signature schemes," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '01. London, UK: Springer-Verlag, 2001, pp. 355–367.

[26] D. Naccache, D. M'Raïhi, S. Vaudenay, and D. Raphaeli, "Can D.S.A. be improved? Complexity trade-offs with the digital signature standard," in *Proceedings of the 13th International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '94)*, 1994, pp. 77–85.

[27] M. Bellare and P. Rogaway, "The exact security of digital signatures: How to sign with RSA and Rabin," in *Proceedings of the 15th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '96)*. Springer-Verlag, 1996, pp. 399–416.

[28] ——, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM conference on Computer and Communications Security (CCS '93)*. NY, USA: ACM, 1993, pp. 62–73.

[29] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," Cambridge, MA, USA, Tech. Rep. MIT/LCS/TR-212, 1979.

[30] A. A. Yavuz, F. Alagöz, and E. Anarim, "HIMUTSIS: Hierarchical multi-tier adaptive ad-hoc network security protocol based on signcryption type key exchange schemes," in *Proceedings of the 21th International Symposium Computer and Information Sciences (ISCIS '06)*, ser. Lecture Notes in Computer Science, vol. 4263. Springer-Verlag, 2006, pp. 434–444.

[31] E. Mykletun, M. Narasimha, and G. Tsudik, "Authentication and integrity in outsourced databases," *Transaction on Storage (TOS)*, vol. 2, no. 2, pp. 107–138, 2006.

[32] National Institute of Standards and Technology, "Federal information processing standard 186: Digital signature standard," http://csrc.nist.gov/publications/, 1993.

[33] S. Chang, W. W. Shieh, S., and C. Hsieh, "An efficient broadcast authentication scheme in wireless sensor networks," in *Proceedings of the ACM Symposium on Information, computer and communications security*, ser. ASIACCS '06. New York, NY, USA: ACM, 2006, pp. 311–320.

[34] M. Bellare and P. Rogaway, "Introduction to modern cryptography," in *UCSD CSE Course*, 1st ed., 2005, p. 207, http://www.cs.ucsd.edu/~mihir/cse207/classnotes.html.

[35] E. Mykletun, M. Narasimha, and G. Tsudik, "Signature bouquets: Immutability for aggregated/condensed signatures," in *Proceedings of the 9th European Symposium on Research in Computer Security (ESORICS '04)*. Springer-Verlag, September 2004, pp. 160–176.

[36] S. Even, O. Goldreich, and S. Micali, "Online/offline digital signatures," in *Proceedings on Advances in Cryptology (CRYPTO '89)*. Springer-Verlag, 1989, pp. 263–275.

[37] Shamus, "Multiprecision integer and rational arithmetic c/c++ library (MIRACL)," http://www.certivox.com/miracl/miracl-download/, Last Accessed on 09/02/2014.

Attila Altay Yavuz received a BS degree in Computer Engineering from Yildiz Technical University (2004) and a MS degree in Computer Science from Bogazici University (2006), both in Istanbul, Turkey. He received his PhD degree in Computer Science from North Carolina State University in August 2011. Between December 2011 and July 2014, he was a member of the security and privacy research group at the Robert Bosch Research and Technology Center North America. Since August 2014, he has been an Assistant Professor in the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, USA. He is also an adjunct faculty at the University of Pittsburgh's School of Information Sciences since January 2013.

Attila Altay Yavuz is interested in design, analysis and application of cryptographic tools and protocols to enhance the security of computer networks and systems. His current research focuses on the following topics: Privacy enhancing technologies (e.g., searchable encryption), security in cloud computing, authentication and integrity mechanisms for resource-constrained devices and large-distributed systems, efficient cryptographic protocols for wireless sensor networks.