Enhancing Demand Response Signal Verification in Automated Demand Response Systems

Daisuke Mashima, Ulrich Herberg, Wei-Peng Chen

Fujitsu Laboratories of America Inc.
Sunnyvale, CA
{dmashima, uherberg, wchen}@us.fujitsu.com

***Abstract*—Demand response (DR) is a promising technology for meeting the world's ever increasing energy demands without a corresponding increase in energy generation, and for providing a sustainable alternative for integrating renewables into the power grid. As a result, interest in automated DR is increasing globally and has led to the development of OpenADR, an internationally-recognized standard in the area. In this paper, we propose security-enhancement mechanisms to provide DR participants with verifiable information that they can use to make informed decisions about the validity of DR signals.**

***Index Terms*—Power System Security, Privacy**

I. INTRODUCTION

OpenADR2.0 standards [1] are being developed by the OpenADR Alliance to define a standardized communication model for automated demand response, including the messaging scheme used between a demand response (DR) automation server, for example at a utility, and DR participants. In recent years, DR automation servers are often operated by DR aggregation service providers (also called DR aggregators), which mediate the interactions and transactions between the utility company and DR participants.

The security scheme implemented in OpenADR largely relies on well-established standards, such as TLS (Transport Layer Security) that ensures sender authenticity, message integrity, and confidentiality and XML Signature for non-repudiation. While such mechanisms are effective in securing communication between a certain pair of nodes, they are not sufficient to attain end-to-end security in communication involving multiple hops (i.e., sequential pairs of nodes), which is often the case when DR aggregators are involved.

In this work, we propose a mechanism to enhance security in multi-hop automated demand response communication that is compliant with the OpenADR2.0 protocol and schema. The proposed mechanism can be immediately implemented in any OpenADR2.0-based systems without affecting compatibilities. Our scheme allows end DR participants to reliably verify which entities are involved in the distribution path of DR event information (also called a DR signal or a DR event signal) and also the semantic consistency between a DR signal issued by the originating entity (e.g., utility) and the DR signal received from an intermediate entity (e.g., a DR aggregator). OpenADR enhanced with the proposed scheme can better protect DR participants and the entire power grid from being manipulated by malicious DR signals that may be inserted into the multi-hop DR signal distribution path by external attackers as well as compromised or dishonest intermediaries.

The rest of the paper is organized as follows. In Section II, an overview of OpenADR2.0 communication model is provided, followed by a discussion of potential security issues. The proposed approach and design are described in Section III. We discuss related work in Section IV and conclude the paper with future work in Section V.

II. OPENADR2.0 OVERVIEW

In this section, an overview of the OpenADR2.0 standards, [1], is provided. OpenADR defines services that are subset of OASIS Energy Interoperation (EI) Version 1.0 [2] for automated demand response. The OpenADR2.0a profile defines the feature set for simple devices (e.g., thermostats), including *EiEvent* service used for distributing DR signals, while the latest 2.0b profile is designed for full-feature energy management solutions including DR aggregation.

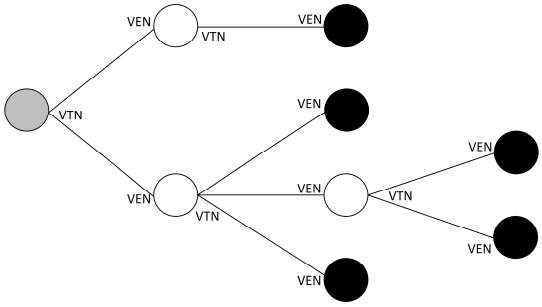

Fig.1: Communication Model in OpenADR

In the communication model of OpenADR (and also of OASIS EI), nodes are organized in a tree-like structure and are categorized into Virtual Top Nodes (*VTN*s) or Virtual End

Nodes (*VEN*s). Some entities, such as DR aggregators, may include both VTN and VEN functionalities. An example of such architecture is shown in Fig. 1. In this work, the left-most node in the figure is called a *top-most VTN*, black nodes in the same figure are called *end-most VEN*s, and the other nodes shown in white are called *intermediaries*.

DR signals are sent via the *EiEvent* service, and *oadrDistributeEvent* payload sent by a VTN conveys the DR events. Note that all messages used in OpenADR2.0 are defined by an XML schema and are transferred over HTTP or XMPP. In an *oadrDistributeEvent* payload, there can be zero or multiple DR event signals, each of which is represented as *eiEvent* element. *eiEvent* contains a number of sub elements, such as identifiers of intended DR event targets (*eiTarget*). Among the sub elements, this paper focuses on the element that is most relevant to the proposed design, namely *eventDescriptor*. As the name implies, this element contains explanatory metadata about the DR event, such as the unique identifier of the event, event creation timestamp, current status of the event, and so forth. In addition, there are a couple of optional elements, including *vtnComment* element that can contain arbitrary text data provided by the VTN for the VEN.

Regarding security, OpenADR2.0 defines only minimal features for hop-to-hop message confidentiality and integrity [1]. There are two security levels: Standard Security and High Security, and implementation of Standard Security is mandated. Standard Security just requires use of TLS, and client certificates have to be used for mutual authentication. High Security additionally requires use of XML Signature. Because of these requirements, we can assume that each node is equipped with a valid public / private key pair and a digital certificate issued by a trusted certification authority (CA).

While the confidentiality and message integrity between two communicating parties, i.e., a VTN and VEN, is assured, such a hop-to-hop security mechanism may not be sufficient in cases where there exist one or multiple intermediaries between a top-most VTN and an end-most VEN. There may be a number of possible threats, but in this work, we consider ones caused by compromised or malicious intermediaries.

First, an end-most VEN can only know and verify a message that is sent by its immediate parent. Although the VEN can verify the message integrity and sender identity through TLS, the VEN cannot verify the consistency between the received message and the message that was originally issued by the top-most VTN. In a typical demand response scenario, an intermediary, such as a DR aggregation service provider, receives a DR event signals from its VTN, processes the signal based on its own business logic, and eventually issues another DR signal for each of its VENs. Therefore, in normal situations, the messages sent to VENs must be semantically consistent with the original DR signal. However, without knowing the DR signal issued by the top-most VTN, there is no way for VENs to detect anomalous messages that are not consistent with the original DR message.

Secondly, the OpenADR communication model implicitly assumes that all nodes in the upstream implement sufficient security protection and appropriately conduct verification of sender identity as well as message integrity. However, such an assumption is not often realistic in the setting where multiple heterogeneous organizations are involved. Lack of rigorous verification would open the door to attackers who attempt to insert bogus DR signals into a path between a top-most VTN and an end-most VEN.

These problems could make VENs vulnerable against attacks. Namely, when a server working as an intermediary is compromised by malware and thereby manipulated by an attacker, it could try to send bogus, malicious DR signals to nodes in its downstream, potentially causing harm to the end-customers, or even worse, the stability of the grid. Considering the tree-like structure of OpenADR communication model, compromise of nodes closer to the root would cause a broad, significant damage in the grid. Moreover, if a VEN deployed in an enterprise, such as a data center, is the target of an attack, it could pose monetary damage on the company. Similar issues could occur when a disgruntled insider of the organization running the intermediary intentionally transmits malicious DR signals to VENs.

III. Enhanced Verification of OpenADR Signals

A. High-level Design of the Proposed Solution

In this section, we present the approach to address the security issues discussed in the previous section. The proposed approach takes advantage of public-key infrastructure, which is already part of the requirements in OpenADR [1].

Consider the situation where two nodes, called I_1 and I_2, mediate the DR signal transmission between a top-most VTN, T, and an end-most VEN, E. We assume that E knows (and can trust) at least T's digital certificate. T is usually a utility company that can be naturally trusted, so this assumption practically holds. The overview is shown in Fig. 2. $[data]_{entity}$ represents that *data* is signed with *entity*'s private key.

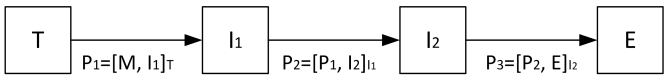

Fig.2: Verifiable DR Signal Distribution Path

When T issues a DR event signal, it also generates a *tag*, which contains metadata (M) of the DR signal, which uniquely identifies and is tied to the DR event signal, e.g., a cryptographic hash value calculated by using the SHA-256 algorithm, and the recipient's identity (I_1), which can be a digital certificate, certificate fingerprint (usually constructed based on a cryptographic hash value of the certificate), or any type of unique identifier of the recipient and to be used by the recipient to retrieve the appropriate public key. The tag is then signed by T, resulting in P_1. P_1 is handed to I_1 along with T's DR event signal corresponding to M, which we call DR_t hereafter, and I_1's identity. I_1 can then specify the next recipient, I_2, and signs the pair of P_1 and I_2's identity. The generated tag, called P_2, together with P_1, I_1's identity, I_2's identity, and DR_t are sent to I_2. Likewise, I_2 generates a signed tag P_3 and hands it with P_1, P_2, I_1's identity, I_2's identity, E's identity, and DR_t to E.

Although in the figure only one node is depicted at each tier, it is possible that there are multiple nodes. In such a case,

a tag is generated for each recipient. For instance, if there is another end-most VEN E', I_2 needs to prepare a different tag whose destination is set to be E'. We assume that utilities and intermediaries, e.g., DR aggregation service providers, have ample resources and thereby can handle such load. Note that end-most VENs, which may be resource constrained, are only required to verify a single tag per DR signal.

Upon receiving the additional information, E can verify who has been involved in the distribution path as follows.

1. Calculate M based on DR_t.
2. Verify P_1 by using T's public key.
3. Retrieve I_1's public key and use it to verify P_2.
4. Retrieve I_2's public key and use it to verify P_3 and also check if the designated recipient is E itself.

After successful verification at step 2, E can be convinced that I_1 is the recipient intended by T regarding the DR signal represented by M. After that, by verifying P_2, E can learn that P_1 and the corresponding data are received by I_1, which establishes the path {T, I_1}. Then, verification at step 4 allows E to verify the path {T, I_1, I_2, E}. If the chain is incomplete (i.e., the designated recipient at some hop is different from the signer of the next) or if suspicious entities are involved in the chain, E has an option to discard the signal. While we discussed only verification at E, any intermediary can also perform the verification of the path to itself in the same way.

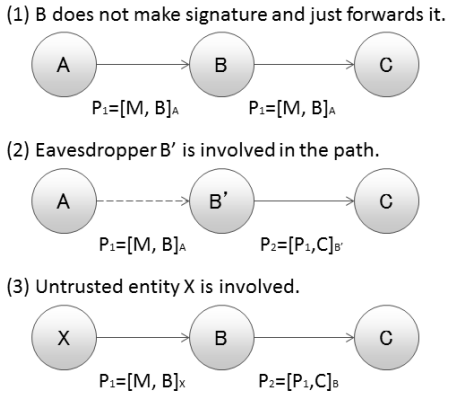

Fig.3: Examples of invalid paths

Fig. 3 illustrates some examples of invalid DR signal distribution paths. In the case of (1), B is designated as the recipient by A and is expected to make signature on P_1. However, C cannot find it and thereby the data can be rejected. In addition, C can notice that the designated recipient is not C itself. (2) is a case where B' intercepts the data and tag sent to B. Even if B' would generate P_2 in the correct way, the recipient specified in P_1 and the signer of the next tag are different, resulting in a disconnected chain. In the example (3), all tags are generated appropriately. However, if C is not aware of or cannot trust X, C can reject the data.

In addition to the path verification discussed above, E can know, through DR_t signed by T, the content of the original DR signal T issued. Each intermediary may issue a different DR signal, which can be denoted as DR_{I1} or DR_{I2}, for entities in its downstream. Such derived signals should have semantic consistency with the original one, as mentioned earlier. Authenticated DR_t also enables E to perform such verification to detect an illegitimate or malicious DR signal issued by an intermediary. Possible validation criteria in case of OpenADR would include: consistency of DR event time and duration, market context, event status (e.g., cancelled or not), level of requested curtailment, and freshness of the DR event.

While we so far discussed that metadata M is constructed as a cryptographic hash value of the DR signal issued by T (i.e., DR_t), forwarding the entire signal may cause a privacy issue. Namely, T may be directly sending the same DR_t to I_3 in addition to I_1. In such a situation, DR_t that E eventually receives may contain I_3's identity even though it is irrelevant to E. Also an intermediary, for the sake of its own privacy, may want to hide some information only related to itself from its customers. To address such cases, it is often desired to allow intermediaries to hide part of DR_t without losing verifiability discussed above. We can employ redactable digital signature schemes [3], instead of regular digital signature schemes, when creating and verifying P_1. For example, T can construct a Merkle Hash Tree [4,5] based on the content of original DR_t and calculate M as the root hash value of the tree. Leaf nodes of a Merkle Hash Tree contain cryptographic hash values of data items. Each non-leaf node is calculated as the hash value of its children nodes, and eventually the tree generates a single root hash value that represents the entire tree, which is then signed. Under this construction, arbitrary data items can be hidden without changing the root hash value, by replacing them with appropriate hash values. In this way, any intermediary can redact arbitrary part of DR_t when necessary, generating a redacted copy of DR_t. It can be later used by any recipient to calculate the same M, and thereby E can verify P_1 and remaining part of the signal distribution path. Therefore, the proposed verification scheme still holds.

B. Implementation in OpenADR2.0 Standards

Based the design discussed in the previous section, in this section we describe a way to enhance security of DR event distribution under OpenADR2.0. As mentioned in Section II, the only place where additional information can be stored is *vtnComment*. Thus, we take advantage of it to be compliant with the standards.

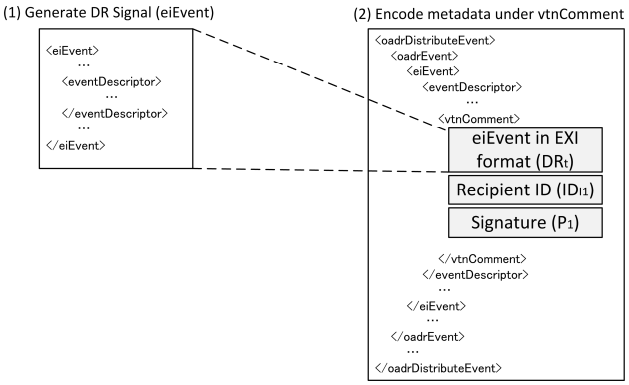

Fig.4: DR signal generation at a top-most VTN

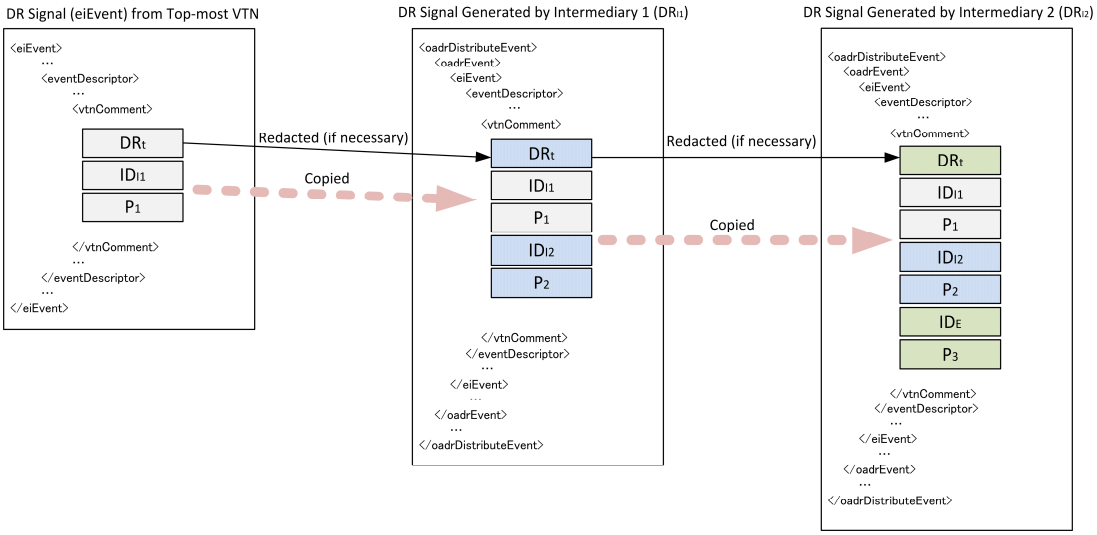

Fig.5: Handling at Intermediaries

A top-most VTN first generates a DR signal as defined in OpenADR2.0. It may include one or multiple *eiEvent* elements. Then, for each *eiEvent*, the top-most VTN first calculates the Merkle root hash value based on the data items contained in the *eiEvent*. Since XML has a tree structure, it can be easily mapped to a Merkle Hash Tree. Note that this root hash value corresponds to M in our design and is used with the recipient's identity (*Recipient ID* in Fig. 4) to create a tag (*Signature* in the figure). Both Recipient ID and Signature are stored in *vtnComment* as Base64-encoded text [7], according to the XML schema of OpenADR. In addition, the original *eiEvent* itself, i.e., DR_t, also needs to be transferred to the downstream entities. We could choose to copy the entire *eiEvent* in the original XML form, but it may cause significant overhead in message size. Use of EXI (Efficient XML Interchange) [6] can mitigate this problem. By using EXI, XML is compressed into binary data. The resulting binary is then encoded into text by using Base64. The resulting text is stored under *vtnComment*. In other words, the compressed form of *eiEvent* content is embedded under its own *vtnComment* along with other metadata. After that, the *eiEvent* is sent to the recipient as part of the *oadrDistributeEvent* payload. These tasks are illustrated in Fig 4.

Note that the signature embedded here is used for different purpose from the regular XML Signature, which can be optionally used in OpenADR2.0, and does not necessarily replace it. The regular XML Signature is basically for hop-to-hop message authentication and non-repudiation whereas our signature aims at establishing verifiable DR-signal distribution path. When XML Signature is used in addition to our proposed scheme, it should be created after the contents of *vtnComment* are prepared.

Handling of DR signals at an intermediary (*Intermediary 1*) that receives a signal directly from a top-most VTN and an intermediary (*Intermediary 2*) at a lower tier is illustrated in Fig. 5. In the following, this is explained using this figure. Since tasks performed by both are almost identical and the only major difference is the length of signature chain to be verified, we here focus on tasks done by Intermediary 1. Also, it is essentially the same even when more intermediaries are involved.

While it is optional and thereby can be skipped, Intermediary 1 can first verify the authenticity of the original *eiEvent* by using the top-most VTN's signature embedded in *vtnComment*. To conduct the tag verification, the intermediary first needs to calculate the Merkle root hash value in the same way as done by the top-most VTN, by using the EXI data. Since EXI compression is invertible, the original XML data can be recovered from DR_t stored in EXI format. Then, the pair of the root hash value and ID_{I1} can be validated against P_1. At this point, if P_1 is not valid or the signer is not a trustworthy entity, the intermediary can discard or reject the DR signal. In addition, in case the specified recipient ID belongs to another party, the DR signal should be considered invalid and should not be accepted.

As the next step, Intermediary 1 generates its own DR signal (DR_{I1}) for its VENs based on DR_t. After that, it embeds, into its own DR signal (more specifically under the *vtnComment* of DR_{I1}), metadata given by the top-most VTN including DR_t. If Intermediary 1 considers that some portion of the top-most VTN's message is privacy sensitive, it can redact the corresponding portion [3]. If redaction is made, DR_t in the left-most box is different from the one in the center box in the figure. However, as discussed in Section III-A, Merkle root hash values calculated from both versions are the same, and thereby the top-most VTN's signature (P_1) is still valid.

After that, Intermediary 1 specifies the ID of its VEN (i.e., Intermediary 2), and then makes signature on the pair of this recipient ID (ID_{I2}) and P_1. Again, they are stored under *vtnComment* of DR_{I1}. The resulting *oadrDistributeEvent* payload conveying DR_{I1} is the one in the middle of Fig. 5.

Finally, let us discuss the verification at an end-most VEN. As defined in OpenADR2.0, a VEN can verify the integrity of the message sent by its immediate VTN. If the VEN is not interested in or is incapable of verifying the distribution path and the contents of the DR signal issued by the top-most VTN, it can just discard the contents of *vtnComment*.

If the VEN is further interested in verifying the information embedded in *vtnComment*, it should follow the similar procedure done by Intermediary 1. As explained in Section III-A, the VEN can eventually be convinced of the integrity of the original DR signal issued by the top-most VTN, which is stored as DR_t, as well as the chain of identities of intermediaries. After such verification, the VEN can use the information to make an informed decision about whether it should accept the received DR signal or not. While the detailed scheme is outside of our scope, VEN could, for example, utilize the criteria mentioned in Section III-A.

If a potentially malicious intermediary redacted too much information for the VEN to make an appropriate decision, the VEN always has freedom to reject the DR signal or can sends back an error code requesting for more information. When the VEN decides to reject the signal for whatever reason, to be compliant with the standard, it should return an error and also opt out from the corresponding DR event.

C. Prototype Implementation

We briefly discuss the prototype implementation in OpenADR-based system including a utility company (i.e., a top-most VTN), two intermediaries emulating DR aggregation service providers, and an end customer (i.e., an end-most VEN). All entities and modules that handle generation and verification of metadata are implemented in Java, and SHA-256 is used as a hash function while 1024-bit RSA is used for digital signature. A commodity laptop PC equipped with dual-core Intel Core i7 processor and 8GB memory is used for the experiments.

Under this setting, we measured processing time added for the proposed scheme to demonstrate the practical aspect. Each measurement below is the average of 10 executions. Generation of metadata by the utility, including calculation of Merkle Hash Tree, signing, and EXI encoding etc., took 23.4ms for each *eiEvent*. The processing time at an intermediary, including redaction of one XML element, update of metadata, was 22.7ms. Distribution path verification at the customer, which involves verification of three digital signatures, took 15ms.

Regarding the communication overhead, based on our prototype, adding the metadata at a top-most VTN, including Base64-encoded EXI data, increases the message size by 50-60% of the original *eiEvent* (usually around 2,500-3,000 Bytes), in case a certificate fingerprint is used to specify a designated recipient. Also 350-400 Bytes are added per hop (i.e., a digital signature and recipient ID). Containing an entire public-key certificate as a recipient identity increases the overhead, but it can eliminate the necessity of certificate lookup at verifiers. Since our extension can be omitted in truly resource-constrained environment, we believe it is still in an acceptable range.

IV. Related Work

Establishment of chain of identity and digital signatures has some similarity to the chain of trust established in typical public-key infrastructure. While it provides verifiable trust relationship starting from the publicly trusted root CA, it is not strictly tied to a specific data item distributed among multiple entities and thereby cannot establish data distribution path.

Our approach is inspired by the scheme designed in [8], which discusses a scheme to establish information accountability on electronic health record (EHR) sharing in a distributed, multi-domain environment. However, in their scheme, metadata attached to data carries only one-hop information and it has to be accumulated on an online repository managed by the data owner. Accumulated metadata can be later used to re-construct the sharing data path. On the other hand, our scheme does not require such a repository.

Ordered Multisignature scheme (OMS) [9], could tell a verifier the order of signers in a verifiable way. One of the limitations of such a scheme is that it is not possible to enforce all intermediaries to create their signatures. In other words, some (possibly malicious) intermediary may choose not to sign and thereby would not be visible to a verifier. On the other hand, under our scheme, the chain will be disconnected in case such intermediaries exist and thereby should be rejected.

V. Conclusions

In this work, we proposed an extension of OpenADR to enhance demand response (DR) participants' capability to verify the validity of received DR event information and thus to protect themselves, as well as the stability of entire power grids, against malicious information emitted by adversaries involved in the DR signal distribution path. Our scheme is designed strictly under the specification of OpenADR2.0 and can be easily integrated into any OpenADR2.0-based systems. We believe that such an enhanced security will contribute to the broader adoption of standards-based auto DR.

One of the future directions is to define an optimal data schema to embed metadata in the OpenADR payload, minimizing communication overhead while balancing the trade-off with computational cost. The scheme discussed in this paper may also be applicable in other contexts where data sharing or distribution among multiple parties is involved, so we will explore such possibilities in our future work.

References

[1] OpenADR Alliance. http://www.openadr.org..

[2] Energy interoperation version 1.0. http://docs.oasis-open.org/energyinterop/ei/v1.0/energyinterop-v1.0.html, 2012.

[3] R. Johnson, D. Molnar, D. X. Song, and D. Wagner. Homomorphic signature schemes. In CT-RSA, pages 244-262, 2002.

[4] R. C. Merkle. Protocols for public key cryptosystems. In IEEE Symposium on Security and Privacy, pages 122-134, 1980.

[5] R. C. Merkle. A certified digital signature. In CRYPTO, pages 218-238, 1989.

[6] J. Schneider and T. Kamiya. Efficient Xml Interchange (EXI) format 1.0. http://www.w3.org/TR/exi/, 2011.

[7] S. Josefsson. The Base16, Base32, and Base64 Data Encodings. http://tools.ietf.org/html/rfc4648, 2006.

[8] D. Mashima and M. Ahamad. Enabling Robust Information Accountability in E-healthcare Systems . In 3rd USENIX Workshop on Health Security and Privacy, 2012.

[9] A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In ACM Conference on Computer and Communications Security, pages 276-285, 2007.