Non-Interactive Editable Signatures for Assured Data Provenance

Haifeng Qian
Department of Computer Science
East China Normal University
haifeng.ecnu@gmail.com

Shouhuai Xu
Department of Computer Science
University of Texas at San Antonio
shxu@cs.utsa.edu

ABSTRACT

In order to make people truly benefit from data sharing, we need technical solutions to assuring the trustworthiness of data received from parties one may not have encountered in the past. Assured data provenance is an important means for this purpose because it (i) allows data providers to get credited for their contribution or sharing of data, (ii) is able to hold the data providers accountable for the data they contributed, and (iii) enables the data providers to supply high-quality data in a *self-healing* fashion. While the above (i) and (ii) have been investigated to some extent, the above (iii) is a new perspective that, to our knowledge, has not been investigated in the literature. In this paper, we introduce a novel cryptographic technique that can simultaneously offer these properties. Our technique is called *editable signatures*, which allow a user, Bob, to edit (e.g., replace, modify, and insert) some portions of the message that is contributed and signed by Alice such that the resulting edited message is jointly signed by Alice and Bob in some fashion. While it is easy to see that the above (i) and (ii) are achieved, the above (iii) is also achieved because Bob may have a better knowledge of the situation that allows him to provide more accurate/trustworthy information than Alice, who may intentionally or unintentionally enter inaccurate or even misleading data into an information network. This is useful because Alice's inaccurate or even misleading information will never be released into an information network if it can be "cleaned" or "healed" by Bob. Specifically, we propose two novel cryptographic constructions that can be used to realize the above functions in some practical settings.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.4.6 [**Security and Protection**]: Authentication

CODASPY'11, February 21–23, 2011, San Antonio, Texas, USA.

General Terms

Security

Keywords

Digital signatures, editable signatures, multisignatures, aggregate signatures, data provenance, assured data provenance, data trustworthiness

1. INTRODUCTION

Because malicious attackers could intentionally enter inaccurate, misleading, or even malicious data into an information network with the aim to influence or manipulate honest people's decision-making, we need technical means to help end users (or data consumers) evaluate the trustworthiness of data received from other parties. Data provenance has a great potential for fulfilling this goal (see, for example, [6, 3, 19, 20, 11, 10]), as long as data provenance information (or provenance data) is adequately protected. Although protecting data provenance is especially important in settings where attacks are possible, it was not until very recently researchers started investigating the security issues related to data provenance [7, 12, 24, 23, 22].

1.1 Assured Data Provenance

While several kinds of data provenance can be relevant in real-life applications, the arguably most important type of provenance information is the so-called *source* provenance [8], which gives the origin of a piece of data. Existing studies on data provenance, such as those mentioned above, mainly focused on *passive data provenance*, meaning that data providers only need to associate the data they contributed with their digital signatures. However, this is not sufficient in many settings. For example, when the private signing key of a user has been compromised but without being promptly detected/revoked, the compromised key may be abused to sign messages that would be treated as trustworthy by receivers because the messages were digitally signed and the signatures can be verified using non-revoked public keys. Moreover, a dishonest user may intentionally abuse the fact — one's private signing key could have been compromised but without being detected — to inject inaccurate/misleading/malicious information into an information network possibly without being detected and punished. In addition, even an honest user, whose private signing key is adequately protected, may unintentionally inject inaccurate information into an information network just because she is unable to observe or produce more precise data.

The aforementioned potential threats against traditional passive data provenance approach inspires us of the following question: How can we achieve *proactive* or *assured* data provenance so that people can truly benefit from the large amount of data available to them? By assured data provenance, we informally mean the following: (i) Data providers can get credited for their contribution or sharing of data, which is important because the providers can have incentives for sharing, possibly higher quality, data. (ii) Data providers can be held accountable for the data they provide, which is important because such accountability can be seen as a deterrence against the introduction of malicious information. (iii) Data providers can jointly "clean" or "heal" the inaccurate, low-quality, misleading, or even malicious information before the relevant data items are released into an information network. Note that we are by-no-means advocating data censorship; in contrast, we believe that the users, who have some relevant data, together might be able to provide more trustworthy data than they individually do. To see the potential value of assured data provenance, let us look at the following example scenario (see also Figure 1).

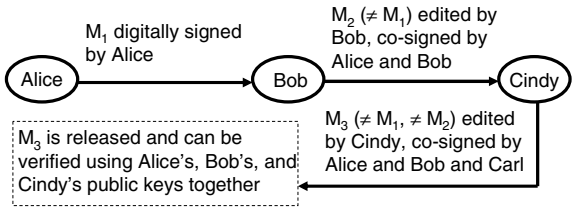

Figure 1: An example of three-party sequentially-editable signature: M_1 is originally signed by Alice, M_2 is edited and "cleaned" by Bob from M_1, M_3 is further edited and "cleaned" by Cindy from M_2. M_3 can be verified using the public keys of Alice, Bob, and Cindy.

Suppose a team of colleagues who need jointly write a report that must be appropriately signed (i.e., each one must be responsible for what he/she wrote in the report) because the report may be used as an input in a decision-making process. More specifically, suppose Alice drafted a message M_1, signed it with her private key, and sent Bob M_1 as well as her digital signature on M_1. Now, Bob may need to partially edit the message M_1 by adding new content and/or modifying some portions of M_1, simply because he has better or more accurate information about the message content (or semantics). Denote by M_2 the resulting new message, which consists of two parts M_{2A} and M_{2B}, where M_{2A} (which can be a proper portion of M_1) was contributed by Alice and M_{2B} is contributed by Bob. How should message M_2 be digitally signed so that the receiver of M_2 can verify that Bob is indeed responsible for the content of M_{2B} and Alice is responsible for the content of M_{2A}? This type of accountability is an important means for enhancing the trustworthiness of M_2. As shown in Figure 1, the above two-party editing chain, namely Alice→Bob, can be naturally extended to multi-party chains (e.g., Alice→Bob→Cindy→ ...).

Assuming that Alice, Bob, and Cindy are honest and that Bob and Cindy do not modify the respective contents of M_1 and M_2 unless he/she is certain about the modifications, M_3 would be more trustworthy than M_2, which in turn would be more trustworthy than M_1. Note that the above argument would be true even if Alice's private signing key has been compromised without being detected because M_1 will be edited by Bob and Cindy as shown in the example. This can be further enhanced by regulating that a message digitally signed by a single user, say M_1 signed by Alice alone, is not deemed as trustworthy as a message that is signed (after editing) by multiple users, correspondingly M_3 in the above example.

1.2 Discussion on Solution Space

Having introduced the usefulness of assured data provenance, let us explore how this problem may be resolved using existing techniques. The first attempt would be to let Bob sign, in addition to M_2, M_1 as well as Alice's signature on M_1. This would allow Cindy to know that M_2 is partially derived from M_1 and additionally contributed by Bob. This approach has several drawbacks and does not quite fulfill the aforementioned assured data provenance because of the following. (i) The resulting communication is the accumulation of the history of M_2. This can be very significant, for example, when M_1 and M_2 have (roughly) the same large size and when there are many signers since the communication cost is proportional to the number of signers. (ii) Cindy has to verify two digital signatures, one by Alice on M_1 and the other by Bob on M_2. In general, the number of digital signatures that need be verified is also proportional to the number of signers, which could be a heavy burden especially when M_1 and M_2 are large (even if they have much content in common). (iii) Even if Cindy may be allowed to known the modifications by Bob, it might be desired in many applications that only the resulting message M_3 is exposed to the outside receivers. The above approach cannot achieve this because the portions of message, which were originally written/signed by Alice and later modified/replaced by Bob and/or Cindy, are also exposed to the outside receivers. One may suggest to use encryption to hide the inner layers of messages and signatures, but this approach incurs extra key-management complexity, which as we will show is unnecessary.

The second attempt would be to let Alice and Bob sign M_2 together, namely that Alice signs M_{2A} and Bob signs M_{2B} because assured data provenance requires accountability. This would require *interaction* between Alice and Bob because Bob needs to inform Alice which part of her message M_1 has been modified by him. This approach would avoid the aforementioned drawbacks of the first attempt because of the following. (i) The resulting communication to Cindy is only M_2 plus its signature and the resulting communication to the outsider receivers is only M_3 plus its signature. (ii) Cryptographic aggregate techniques can "merge" Alice's, Bob's, and Cindy's signatures into a single one. (iii) The receiver cannot infer, for example, the content originally drafted by Alice but later modified by Bob or Cindy as long as the underlying communication channels are private, which is much easier to implement than a full-fledged key management. However, this approach has the very significant drawback that it requires interactions between the signers. For two-party scenarios, it requires one interaction between the two signers. However, for n-party scenarios, it could require $\frac{n(n-1)}{2}$ interactions in the worst case because the content of each signer may be edited/modified by later

ones. The heavy interaction requirement may block the signing process unless all the signers always stay online. (Note that the above discussion also disqualifies the attempt to let the signers interactively edit messages without signing until the final message has been agreed upon. One burden of this naive approach is that the signers have to keep track of "who said what" and may cause disputes given that no one has really "committed" anything before he signs in the final phase.)

The above discussion raises the following question: Can we have a cryptographic mechanism that can avoid all of the drawbacks mentioned above? In this paper, we answer this question affirmatively by presenting a novel cryptographic tool for this purpose.

1.3 Our Contributions

The conceptual contribution of the present paper is the introduction of assured data provenance, which moves a step beyond the current paradigm of passive data provenance. The most important technical contribution of the present paper is the introduction of a new type of digital signatures called *editable signatures*. (Non-interactive) editable signatures are especially useful for assured data provenance when multiple signers need to jointly sign a message, which is the final outcome of some editing process (rather than in the much simpler case when the message is given as input in the beginning of the signing protocol), and might be of independent value.

We present two concrete non-interactive editable signature schemes. The first scheme gives the signers much freedom in the sense that any one can edit any message blocks. The resulting scheme is proven secure in the random oracle model, but it requires $O(n)$ pairing operations to verify such a signature. The second scheme gives the signers less freedom because it restricts which signer can edit which message blocks. Nevertheless, the resulting scheme is proven secure in the standard model (i.e., without using random oracle) while requiring only $O(1)$ pairing operations to verify such a signature.

Paper organization. Section 2 presents definitions of editable signatures. Section 3 describes the cryptographic preliminaries. Sections 4 and 5 present and analyze our two editable signature schemes, respectively. Section 6 briefly discusses previous related work. Section 7 concludes the paper with a discussion on future research directions.

2. DEFINITION AND SECURITY MODEL

2.1 High-level Ideas

Suppose that there are n (sequential) signers $\{A_i | i = 1, \ldots, n\}$, each having a pair of public and private keys (pk_i, sk_i), and that A_1 is the first signer (or the initiator), who drafts the first message M_1 and generates a signature σ_1 on M_1 using its private key sk_1. A_1 sends (M_1, σ_1) possibly through a secure channel to the next signer A_2. (Note that the secure channel can be realized using standard cryptosystems.) Then, A_2 will "edit" message M_1 to obtain a new message M_2 with an accompanying signature σ_2, which is partly based on σ_1. We also call the other signers the editors because they can edit both the messages and signatures provided by previous signers in some fashion. To facilitate message editing, we divide a message into ℓ blocks (chunks), namely $M_i = (m_1, \ldots, m_\ell)$, so that a block m_j may be drafted by one signer and edited by other signers. For the sake of convenience, let us treat a message $M_i = (m_1, \ldots, m_\ell)$ as an ordered set of ℓ elements, with possibly some $m_j = \perp$ where $\perp$ is a placeholder and a special message block (but is not signed in a cryptographical sense as it will become clear later). In other words, M_i is treated as an ordered set $M_i = \{m_1, \ldots, m_\ell\}$ or equivalently $\{(1, m_1), \ldots, (\ell, m_\ell)\}$; these representations will be used interchangeably as it will become clear from the context. Nevertheless, there are three subtle issues.

First, who should decide which signer can sign/edit which message blocks? Note that in the most general case, a signer A_i $(1 \leq i \leq n-1)$ does not have to restrict "which message blocks can be edited by which of the future signers A_j $(i < j \leq n)$." However, it may be desired sometimes that A_i can pre-determine which message blocks may be edited by which signers. This can only be achieved, of course, when the future signers do not sign messages from scratch (otherwise, for example, A_2 can simply disregard σ_1 on message M_1 while signing M_1 using sk_2). This does have practical meanings in assured data provenance because A_2 may not want to be held accountable for the message blocks in M_2 but was provided by M_1, namely the message blocks belonging to $M_1 \cap M_2$, while A_2 is responsible for the message blocks belonging to $M_2 \setminus M_1$. Moreover, in some cases it may be even desired that certain blocks of a final message must be signed by certain pre-defined signers (e.g., such predetermined attribution may be up to the first signer A_1). This allows to introduce some kind of access control that may be needed in some application scenarios.

Second, how should we represent and reflect the editing operations? Suppose the signers eventually generate a signature σ on a message $M_n = (m_1, \ldots, m_\ell)$. To reflect the editing operations, each message block m_i, $1 \leq i \leq \ell$, is associated with Γ_i, which bookkeeps the identities of its signers. In any case, the editing operation implies that each message block is signed at least by one signer, but it is not necessary that every message block is signed by the same set of signers.

Third, how are the (intermediate) signatures verified? The key issue here is, unlike σ_1 that can be verified using pk_1 alone, how σ_2 will be verified. In general, σ_2 might need be verified using both pk_1 and pk_2 because portion of M_2 was actually produced and thus signed by A_1 (and possibly by A_2 as well), and the other portion of M_2 was actually produced and thus signed by A_2 only. As we elaborate below, editable signatures are indeed general and accommodate the very useful notions of multisignatures, which allow multiple signers to endorse the same message (the concept was introduced by [13] and numerous elegant schemes have been proposed, including [4, 1]), and aggregate signatures, which allow to "absorb" multiple signers' signatures on multiple messages into a single one so as to reduce the length of signature tags (the concept was introduced by [4] and numerous elegant schemes have been proposed, including [15, 2, 16]). Specifically, we encounter the following interesting scenarios (using $n = 2$ as an example):

1) A_2 simply inherits whatever A_1 said in M_1. A_2 may sign (some message blocks of) M_2 using its private key sk_2. When A_2 signs M_2, σ_2 is a multisignature on message M_1.

2) A_2 deletes some actual (i.e., non-$\perp$) message blocks in M_1 and without adding actual message blocks (to

replace the deleted message blocks). In this case, the non-$\perp$ blocks of M_2 may be signed by both A_1 and A_2 (i.e., effectively a multisignature), or A_2 only signs some message blocks of M_2. It is challenging to technically allow this because we need to enable A_2 to "cancel" the portion of A_1's signature on $M_1 \setminus M_2$, without giving A_1's private key to A_2.

3) A_2 adds (but not necessarily appends) some actual message blocks to M_1. The resulting signature is an aggregate signature, where the message blocks belonging to M_1 may be signed by both A_1 and A_2 (i.e., effectively a multisignature) and the message blocks belonging to $M_2 \setminus M_1$ are signed by A_2 alone. It is also possible A_2 signs some message blocks of $M_1 \cap M_2$.

4) A_2 completely disagrees with what A_1 said in M_1. In this case, σ_2 is indeed a normal signature on M_2 that can be verified using public key pk_2 alone.

5) A_2 modifies (i.e., not adding or deleting) some message blocks. This is likely the most common scenario in practice, where the message blocks belonging to $M_1 \cap M_2$ are produced and signed by A_1 and possibly signed by A_2 as well (i.e., effectively a multisignature), the original message blocks belonging to $M_1 \setminus (M_1 \cap M_2)$ were produced and signed by A_1 and later edited by A_2, the message blocks belonging to $M_2 \setminus (M_1 \cap M_2)$ are newly produced and signed by A_2, and the resulting signature is an aggregate signature in spirit. Again, it is challenging because we need to enable A_2 to "cancel" the portion of A_1's signature on $M_1 \setminus (M_1 \cap M_2)$.

REMARK 2.1. *Non-interactive editable signatures may have an inherent drawback as we show in the following example. Suppose there are $n = 3$ signers, each message has $\ell = 2$ blocks, A_1 produces $M_1 = (m_1, m_2)$, A_2 edits M_1 to produce $M_2 = (m_1, \perp)$, and A_3 edits M_2 to produce $M_3 = (m_1, m_2)$, which is possible because A_2 thinks m_2 is wrong data but both A_1 and A_3 think m_2 is correct data. In the resulting signature on M_3, m_2 is only signed by A_3 while A_1 actually signed it as well. This may be seen as a kind of loss of information in applications such as evaluating the trustworthiness of M_3. This appears to be inherent to non-interactive editable signatures, and we leave it to future work to resolve this issue.*

2.2 Functional Definition

Suppose n signers $A_1, A_2, \ldots, A_n$ want to jointly edit and sign messages up to ℓ blocks in the fashion discussed above. For the sake of clarification, here we summarize the main notations used in the paper:

- A_i: The i-th signer (or editor if $i \neq 1$) and A_1 is the first signer (initiator), where $1 \leq i \leq n$.
- S: The set of all indices of message blocks (i.e., $S = \{1, \ldots, \ell\}$).
- $M = (m_1, \ldots, m_\ell)$ and $M' = (m'_1, \ldots, m'_\ell)$: M is the message A_i ($1 < i \leq n$) received from signer A_{i-1}. M' is the message sent by signer A_i to signer A_{i+1} when $1 < i < n$, and is the final message when $i = n$ (in this case, the final message M' and its signature are released into an information network).
- Δ: The set of indices of message blocks that can be edited by an editor.
- $C \subseteq S$, $I \subseteq C$ and $\overline{I} \subseteq \Delta$: Suppose A_i ($1 < i \leq n$) receives $M = (m_1, \ldots, m_\ell)$ and outputs $M' = (m'_1, \ldots, m'_\ell)$. Then C is the set of indices of message blocks that A_i does not edit, no matter A_i is allowed to edit or not. In other words, $C = \{i | m_i = m'_i, 1 \leq i \leq \ell\}$.

  I is the set of indices of message blocks that A_i does sign (because, for example, A_i is certain about the content of these message blocks). Note that it is possible that $I \setminus \Delta \neq \emptyset$, meaning that A_i can sign the message blocks A_i is not allowed to edit.

  $\overline{I}$ is the set of indices of message blocks that A_i indeed edited. Note that $\overline{I} = \Delta \setminus (C \cap \Delta) = \Delta \setminus C$.
- $\mathsf{edinfo}_{i-1} = (K_\Delta, \Delta)$ and $\mathsf{edinfo}_i = (K_{\Delta'}, \Delta')$, where $1 < i \leq n$: In the case $1 < i < n$, editor A_i uses the input auxiliary information $\mathsf{edinfo}_{i-1} = (K_\Delta, \Delta)$ to edit the message blocks in message M_{i-1} as specified by Δ, where K_Δ is the set of "editing keys" needed for editing signature σ_{i-1} provided by A_{i-1} with respect to Δ. Suppose A_i determines to allow the next editor A_{i+1} to edit message blocks as specified by Δ'. Then A_i will also prepare auxiliary information $\mathsf{edinfo}_i = (K_{\Delta'}, \Delta')$ for next editor A_{i+1}, who can then use $K_{\Delta'}$ to edit the message blocks in message M_i as specified by Δ'.

  In the case $i = n$, editor $A_i = A_n$ does the same as the above, except that it does not prepare edinfo_i as it is the last editor.
- Γ_i: The set of indices of message blocks (including possibly $\perp$ blocks) signed by A_i.
- L: The list of pairs (pk_i, Γ_i), which allows to correctly indicate "who signed which actual message blocks" and allows to verify signatures.

The functional definition of (sequential but) non-interactive editable signatures is described as follows.

DEFINITION 2.2. *A non-interactive editable signature scheme* $\mathsf{ES} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{FSign}, \mathsf{ESign}, \mathsf{Vf})$ *consists of the following algorithms or protocols:*

$\mathsf{Setup}(1^\lambda)$**:** *A randomized algorithm (possibly run by a central authority) that takes as input a security parameter λ, produces a set of global public parameters pp, and possibly specifies which signers have a final say on which message blocks (e.g., based on some policy).*

$\mathsf{Gen}(pp)$**:** *A probabilistic algorithm that, on input of public parameters pp, outputs an honest signer's pair of private and public keys (sk, pk).*

$\mathsf{FSign}(pp, sk_1, M_1)$**:** *Assume A_1 is the first signer with a pair of private and public keys (sk_1, pk_1). Given pp, $M_1 = (m_1, \ldots, m_\ell)$ with possibly some $m_i = \perp$, A_1 uses its private key sk_1 to generate a signature σ_1 on M_1, chooses a set $\Delta \subseteq S$, generates auxiliary information $\mathsf{edinfo}_1 = (K_\Delta, \Delta)$ which may be optional but otherwise allows the others to edit M_1 and σ_1 properly, and sets $L_1 = (pk_1, \Gamma_1)$ where $\Gamma_1 \subseteq S$ indicates the (actual) message blocks A_1 signed. Finally, it outputs $(M_1, \sigma_1, L_1, \mathsf{edinfo}_1)$, which will be sent to the second signer A_2 possibly over a private channel.*

$\mathsf{ESign}(pp, sk_i, M_i, M_{i-1}, \sigma_{i-1}, L_{i-1}, \mathsf{edinfo}_{i-1})$ **for** $1 < i \leq n$**:** *Given the public parameters pp, M_{i-1}, the received σ_{i-1}, $L_{i-1} = [(pk_1, \Gamma_1), \ldots, (pk_{i-1}, \Gamma_{i-1})]$, and the resulting message M_i after A_i edits M_{i-1}, signer A_i first checks the validity of $(M_{i-1}, \sigma_{i-1}, L_{i-1})$ via the following algorithm $\mathsf{Vf}(pp, M_{i-1}, \sigma_{i-1}, L_{i-1})$. If invalid, A_i rejects and aborts; otherwise, A_i uses its private key sk_i and the received auxiliary information $\mathsf{edinfo}_{i-1} = (K_\Delta, \Delta)$ to edit the received signature σ_{i-1} to produce signature σ_i, prepares $(M_i, \sigma_i, L_i, \mathsf{edinfo}_i)$ where $L_i = [(pk_1, \Gamma_1'), \ldots, (pk_i, \Gamma_i')]$ and $\mathsf{edinfo}_i = \mathtt{null}$ if $i \geq n$ and $\mathsf{edinfo}_i = (K_{\Delta'}, \Delta')$ otherwise. If $i < n$, A_i sends $(M_i, \sigma_i, L_i, \mathsf{edinfo}_i)$ to signer A_{i+1}; otherwise, (M_i, σ_i, L_i) is the final message as well as its signature (that can be released to some information network). Note that Δ' may be determined by A_i according to some policy, and $K_{\Delta'}$ may be derived from edinfo_{i-1} and A_i's own private information. Note also that edinfo_i may be optional.*

$\mathsf{Vf}(pp, M, \sigma, L)$**:** *Given parameters pp, L (the set of signers' public keys), a message M and an alleged signature σ, this deterministic algorithm outputs* 0 *(reject) or* 1 *(accept).*

We require an editable signature scheme to be *correct*, meaning that if all signers follow the protocols, then all of the resulting signatures will always be accepted as valid.

2.3 Security Model

Informally, security of non-interactive editable signature schemes requires that it is infeasible for an attacker to forge an editable signature involving at least one honest signer. Without loss of generality, we assume there is a single honest signer. The adversary can corrupt the other signers, and can choose their keys arbitrarily (but we require users to prove knowledge of their private keys during public-key registration with a Certification Authority or CA). More specifically, we require the adversary to hand all the private keys of the compromised signers to the CA, meaning that our scheme operates in the so-called Knowledge of Secret Key (KOSK) model [2]; we leave it to future work to weaken the operational model [17] as will be discussed in Section 7. Security definition will be split into two scenarios depending on who will be the target of attack: the first signer (called INITIATOR UNFORGEABILITY) or a future signer (called EDITOR UNFORGEABILITY). Both definitions are extensions to the classical security notion of digital signatures [9].

For INITIATOR UNFORGEABILITY, we require that no attacker can have a non-negligible advantage in the experiment specified in Figure 2, where the advantage of adversary $\mathcal{A}$ against $\mathsf{ES} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{FSign}, \mathsf{ESign}, \mathsf{Vf})$ is defined as the probability that experiment $\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{iu.cma}}(\mathcal{A})$ outputs 1.

DEFINITION 2.3 (INITIATOR UNFORGEABILITY). *We say that an adversary can (t, ε_1)-break the* INITIATOR UNFORGEABILITY *of* ES*, if it in time t, has an advantage more than ε_1 (i.e., $\Pr[\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{iu.cma}}(\mathcal{A}) = 1] \geq \varepsilon_1$). If there are no such adversaries, we say the scheme is (t, ε_1)-initiator-unforgeable.*

For EDITOR UNFORGEABILITY, we require that no attacker can gain a non-negligible advantage in the experiment specified in Figure 3, where the advantage of adversary $\mathcal{A}$ against $\mathsf{ES} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{FSign}, \mathsf{ESign}, \mathsf{Vf})$ is defined as the probability that experiment $\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{eu.cma}}(\mathcal{A})$ outputs 1.

Experiment $\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{iu.cma}}(\mathcal{A})$:

1. $pp \leftarrow \mathsf{Setup}(1^\lambda)$; $(pk^\star, sk^\star) \leftarrow \mathsf{Gen}(pp)$; $\mathcal{M} \leftarrow \phi$.
2. Run $\mathcal{A}(pp, pk^\star)$ with $pk_1 = pk^\star$ by handling oracle queries as follows:
   - To query key registration oracle, if a user wants to register pk with corresponding sk, the oracle requires sk. If sk is valid with respect to pk, the oracle returns (pk, c_{pk}), where c_{pk} is the public key certificate. We use a list consisting of $\mathsf{certList} = \{(pk, c_{pk}, sk)\}$ to record the registered keys.
   - To query the FSign oracle, $\mathcal{A}$ prepares $\Delta \subseteq S = \{1, \ldots, \ell\}$ and message M. $\mathcal{A}$ can initiate the FSign oracle concurrently and can interact with "clones" of the honest signer, where each clone maintains its own state, uses its own coins and the keys $(pk^\star, sk^\star)$ and then outputs M, σ, $\mathsf{edinfo} = (K_\Delta, \Delta)$ and $L = (pk_1, S)$. Bookkeep previously signed message blocks in $\mathcal{M}$.
   - To query the hash oracle, $\mathcal{A}$ submits a string and obtains its corresponding value from a random oracle (this oracle is optional if the scheme is constructed in the standard model).
   - Finally, $\mathcal{A}$ outputs a signature (M, σ, L), where $M = (m_1, \ldots, m_\ell)$.
3. If $(\mathsf{Vf}(pp, M, L) = 1) \wedge (pk^\star \in L[1]) \wedge m_i \notin \mathcal{M} = 1$ where $m_i \in M$ is signed by $sk^\star$ and $L[1] \setminus \{pk^\star\} \subset \mathsf{certList}[1]$, then return 1, otherwise return 0.

   Note that $L[1]$ and $\mathsf{certList}[1]$ represent the sets of the first coordinates of L and certList, respectively.

Figure 2: INITIATOR-UNFORGEABILITY **Experiment**

DEFINITION 2.4 (EDITOR UNFORGEABILITY). *We say that an adversary can (t, ε_2)-break the* EDITOR UNFORGEABILITY *of* ES*, if it in time t, has an advantage more than ε_2 (i.e., $\Pr[\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{eu.cma}}(\mathcal{A}) = 1] \geq \varepsilon_2$). If there are no such adversaries, we say the scheme is (t, ε_2)-editor-unforgeable.*

3. PRELIMINARIES

Cryptographic setting and assumption. In this section we briefly review some standard cryptographic setting [5]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicative) cyclic groups of prime order p where the group actions on $\mathbb{G}$, $\mathbb{G}_T$ can be computed efficiently, g be a generator of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be an efficiently computable map with the following properties:

- ⋆ Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$;
- ⋆ Non-degenerate: $e(g, g) \neq 1$.

We say $\mathbb{G}$ is a bilinear group if it satisfies these requirements.

DEFINITION 3.1. ([5]) *In a bilinear group $\mathbb{G}$, the Compu-*

Experiment $\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{eu.cma}}(\mathcal{A})$:

1. $pp \leftarrow \mathsf{Setup}(1^\lambda)$; $(pk^\star, sk^\star) \leftarrow \mathsf{Gen}(pp)$; $\mathcal{M} \leftarrow \phi$. Without loss of generality, we assume $(pk^\star, sk^\star) = (pk_i, sk_i)$ where $i > 1$.
2. Run $\mathcal{A}(pp, pk^\star)$ by handling oracle queries as follows:
   - To query key registration oracle, if a user wants to register a public key pk corresponding to private key sk, the oracle requires sk. If sk is valid with respect to pk, the oracle returns (pk, c_{pk}) where c_{pk} is the public key certificate. We use a list consisting of $\mathsf{certList} = \{(pk, c_{pk}, sk)\}$ to record the registered keys.
   - To query the ESign oracle, $\mathcal{A}$ prepares M', (M, σ, L), and $\mathsf{edinfo} = (K_\Delta, \Delta)$. $\mathcal{A}$ can initiate the ESign oracle concurrently and can interact with "clones" of the honest signer, where each clone maintains its own state, uses its own coins and the keys $(pk^\star, sk^\star)$ and then outputs $(M', \sigma', L', \mathsf{edinfo}')$. Bookkeep previously signed blocks are included in $\mathcal{M}$.
   - To query the hash oracle, $\mathcal{A}$ submits a string and obtains its corresponding value from random oracle (this oracle is optional if the scheme operates in the standard model).
   - Finally, $\mathcal{A}$ outputs a signature (M, σ, L), where $M = (m_1, \ldots, m_\ell)$.
3. If $(\mathsf{Vf}(pp, M, L) = 1) \wedge (pk^\star \in L[1]) \wedge m_i \notin \mathcal{M}) = 1$ where $m_i \in M$ is signed by $sk^\star$ and $L[1] \setminus \{pk^\star\} \subset \mathsf{certList}[1]$, then return 1; otherwise, return 0.

Figure 3: EDITOR-UNFORGEABILITY **Experiment**

tational Diffie-Hellman (CDH) problem is: given $(g, g^a, g^b) \in \mathbb{G}^3$ *for some* $a, b \overset{R}{\leftarrow} \mathbb{Z}_p$*, find* $g^{ab} \in \mathbb{G}$.

Define the success probability of an algorithm $\mathcal{A}$ in solving the CDH problem on $\mathbb{G}$ as

$$\mathrm{Adv}^{\mathrm{cdh}}_{\mathcal{A}} \overset{\mathrm{def}}{=} \Pr\left[g^{ab} \leftarrow \mathcal{A}(g, g^a, g^b) : a, b \overset{R}{\leftarrow} \mathbb{Z}_p\right].$$

The probability is taken over the random choice of g from $\mathbb{G}$, of a, b from $\mathbb{Z}_p$, and the coin tosses of $\mathcal{A}$. We say an algorithm $\mathcal{A}$ (t, ε)-breaks the CDH problem on $\mathbb{G}$ if $\mathcal{A}$ runs in time at most t and $\mathrm{Adv}^{\mathrm{cdh}}_{\mathcal{A}} \geq \varepsilon$. If no adversary $\mathcal{A}$ can (t, ε)-break the CDH problem on $\mathbb{G}$, we say the CDH problem on $\mathbb{G}$ is (t, ε)-secure.

BLS signatures [5]. Since we will use the BLS signature scheme [5] as a starting point for our first scheme (presented in Section 4), we now review the BLS signature scheme $\mathcal{BLS} = (\mathsf{BLS.Gen}(1^\lambda), \mathsf{BLS.Sig}(sk, m), \mathsf{BLS.Ver}_{pk}(m, \sigma))$, which is specified in the afore-mentioned cryptographic setting.

$\mathsf{BLS.Gen}(1^\lambda)$: Pick random $x \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute the public key $pk = g^x$. The private key is $sk = x$. The scheme also needs a random oracle $H : \{0,1\}^* \to \mathbb{G}$.

$\mathsf{BLS.Sig}(sk, m)$: Given private key $sk = x$, and message m, compute and output $\sigma = H(m)^x$ as the signature.

$\mathsf{BLS.Ver}(pk, m, \sigma)$: On input public key $pk = g^x$, message M, and alleged signature σ, verify that

$$e(\sigma, g) \overset{?}{=} e(H(m), pk)$$

holds; if so, output 1 (accept), otherwise output 0 (reject).

It was proven in [5] that the scheme is existentially unforgeable under adaptive chosen-message attack [9] based on the hardness of the CDH problem in the random oracle model.

Waters signatures [21]. Since we will use the Waters signature scheme in [21] as a starting point for our second scheme presented in Section 5, we here briefly review it. Suppose a message is a bit string belonging to $\{0,1\}^k$ for some fixed k (in practice one may first apply a collision-resistant hash function $H' : \{0,1\}^* \to \{0,1\}^k$ to messages of arbitrary length). The scheme uses, in the afore-mentioned cryptographic setting, random generators $g, d \in \mathbb{G}$ and a vector of another $k+1$ random elements $\mu = (u', u_1, \ldots, u_k) \in \mathbb{G}^{k+1}$, where $u', u_1, \ldots, u_k$ define a function $H(\cdot)$ that given $m = (b_1, \ldots, b_k) \in \{0,1\}^k$, maps m to $H(m) = u' \prod_{i=1}^{k} u_i^{b_i} \in \mathbb{G}$. The scheme $\mathcal{WS} = (\mathsf{W.Gen}(1^\lambda), \mathsf{W.Sig}(sk, m), \mathsf{W.Ver}(pk, m, \sigma))$ is specified as follows.

$\mathsf{W.Gen}(1^\lambda)$: Pick $x \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute $B \leftarrow e(h, g^x)$. The public key is $pk = (\mu, B, d, g)$ and the private key is $sk = d^x$.

$\mathsf{W.Sig}(sk, m)$: Given sk and message $m = (b_1, \ldots, b_k) \in \{0,1\}^k$, pick a random $r \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute

$$s \leftarrow d^x \cdot H(m)^r \quad \text{and} \quad t \leftarrow g^r.$$

The signature is $\sigma = (s, t) \in \mathbb{G}^2$.

$\mathsf{W.Ver}(pk, m, \sigma)$: Given public key pk, $m = (b_1, \ldots, b_k) \in \{0,1\}^k$, and $\sigma = (s, t) \in \mathbb{G}^2$, verify that

$$e(s, g) \overset{?}{=} A \cdot e\,(t, H(m))$$

holds; if so, output 1 (accept), otherwise output 0 (reject).

It was proven in [21] that the scheme is existentially unforgeable under adaptive chosen-message attack [9] based on the hardness of the CDH problem without using random oracles.

4. FULLY EDITABLE SIGNATURES

In this section we present a non-interactive fully-editable signature scheme, which gives the editors much freedom in terms of the message blocks they can edit. Specifically, in such a scheme an editor can edit the received message by itself and can assign to other editors the capability of further editing. Let $H : \{0,1\}^* \to \mathbb{G}$ be a hash function (random oracle). For $1 \leq i \leq \ell$, define

$$\mathsf{H}(i||m_i||pk_1) = \begin{cases} 1, & \text{if } m_i = \perp \\ H(i||m_i||pk_1), & \text{if } m_i \neq \perp. \end{cases}$$

The scheme is based on the afore-mentioned BLS signature scheme [5].

4.1 Construction

Recall that there are n signers $A_1, \ldots, A_n$; $S = \{1, \ldots, \ell\}$; $M = (m_1, \ldots, m_\ell)$; $\Gamma_i \subseteq S$ is the set of indices of message blocks that are signed by signer A_i. The scheme is described as follows.

$\underline{\mathsf{Setup}(1^\lambda)}$**:** On input a security parameter λ, it generates $\mathbb{G}$, $\mathbb{G}_T$, p, g, e, and random oracle H as specified above. Let $pp = (\mathbb{G}, \mathbb{G}_T, p, g, e, H)$, which is made public.

$\underline{\mathsf{Gen}(pp)}$**:** It takes pp as input, randomly chooses $x_i \xleftarrow{R} \mathbb{Z}_p$, outputs signer A_i's pair of private and public keys $(sk = x_i, pk = g^{x_i})$.

$\underline{\mathsf{FSign}(pp, sk_1, M_1)}$**:** On input $M_1 = (m_1, \ldots, m_\ell)$, A_1 determines a set $\Delta \subseteq S$ (the set of indices of blocks of M_1 that can be edited by signer A_2, including the $\perp$ blocks[1]), uses $sk_1 = x_1$ to compute

$$\sigma_1 = \left(\prod_{i=1}^{\ell} \mathsf{H}(i||m_i||pk_1) \right)^{x_1},$$

sets $s_\alpha = \mathsf{H}(\alpha||m_\alpha||pk_1)^{x_1}$ for $\alpha \in \Delta$, $K_\Delta = \{s_\alpha\}$, $\Gamma_1 = S$ (meaning that A_1 signed all the ℓ blocks, possibly including $\perp$ blocks), $\mathsf{edinfo}_1 = (K_\Delta, \Delta)$, and $L_1 = [(pk_1, \Gamma_1)]$. Finally A_1 sends $(M_1, \sigma_1, L_1, \mathsf{edinfo}_1)$ to signer A_2 over a private channel (which can be implemented using standard cryptosystems).

$\underline{\mathsf{ESign}(pp, sk_i, M', M, \sigma, L, \mathsf{edinfo})}$**:** Parse $M = (m_1, \ldots, m_\ell)$, $L = [(pk_1, \Gamma_1), \ldots, (pk_{i-1}, \Gamma_{i-1})]$ where Γ_j $(1 \le j \le i-1)$ is the set of the indices of the message blocks (including possibly $\perp$ blocks) signed by A_i, $M' = (m'_1, \ldots, m'_\ell)$ and $\mathsf{edinfo} = (K_\Delta, \Delta)$, where $K_\Delta = \{s_\alpha | \alpha \in \Delta\}$ (herein $\bigcup_{j=1}^{i-1} \Gamma_j = S$). If

$$e(\sigma, g) \neq \prod_{j=1}^{i-1} e\left(\prod_{\beta \in \Gamma_j} \mathsf{H}(\beta||m_\beta||pk_j), pk_j \right),$$

abort; otherwise execute as follows:

- In the case $1 < i < n$, execute the following:
  1. Let $C = \{\alpha | m'_\alpha = m_\alpha, \alpha \in S\}$ be the set of indices of message blocks that A_i inherits (i.e., copy-and-paste) from A_{i-1}. Note that A_i does not have to sign those blocks with respect to C because, for example, A_i may be uncertain about the trustworthiness of these message blocks.
  2. Choose $I \subseteq C$ where I is the set of indices of messages blocks that A_i inherits and will sign as well. Let $\overline{I} = \Delta \setminus (C \cap \Delta) = \Delta \setminus C$, which is the set of indices of message blocks A_i will edit and sign.
  3. Compute

$$\sigma' = \sigma \cdot \left(\prod_{\alpha \in I \cup \overline{I}} \mathsf{H}(\alpha||m'_\alpha||pk_i) \right)^{x_i} \cdot \prod_{\alpha \in \overline{I}} s_\alpha^{-1}.$$

  4. Update $\Gamma_j = \Gamma_j \setminus \overline{I}$ for $j = 1, \ldots, i-1$ and set $\Gamma_i = I \cup \overline{I}$.
  5. Determine $\Delta' \subseteq \Delta$ according to some policy.
  6. Update

$$s'_\alpha = \begin{cases} \mathsf{H}(\alpha||m'_\alpha||pk_i)^{x_i}, & \text{for } \alpha \in \overline{I} \bigcap \Delta' \\ s_\alpha \cdot \mathsf{H}(\alpha||m'_\alpha||pk_i)^{x_i}, & \text{for } \alpha \in I \bigcap \Delta' \\ s_\alpha, & \text{for } \alpha \in (C \setminus I) \bigcap \Delta' \end{cases}$$

  and let

$$K_{\Delta'} = \{s'_\alpha | \alpha \in \Delta'\}.$$

  7. Set $L' = [(pk_1, \Gamma_1), \ldots, (pk_i, \Gamma_i)]$ and return

$$(M', \sigma', L', \mathsf{edinfo}')$$

  where $\mathsf{edinfo}' = (K_{\Delta'}, \Delta')$.
- In the case $i = n$, namely that the signer is the last one, the signer executes the same as A_j $(j \neq n)$ does, except that it sets $\mathsf{edinfo}' = \texttt{null}$. The final signature output is (M', σ', L').

$\underline{\mathsf{Vf}(pp, M, \sigma, L)}$**:** Given parameters pp,

$$L = [(pk_1, \Gamma_1), \ldots, (pk_n, \Gamma_n)], \quad M = (m_1, \ldots, m_\ell)$$

and an alleged signature σ, the verifier accepts the signature if

$$e(\sigma, g) = \prod_{j=1}^{n} e\left(\prod_{\beta \in \Gamma_j} \mathsf{H}(\beta||m_\beta||pk_j), pk_j \right), \quad (4.1)$$

and reject otherwise.

We stress that the channel between the signers are private, which is important because the auxiliary information edinfo should be kept secret to the respective pair of signers.

[1]This is for the sake of convenience in specifying the scheme. Technically, A_2 can edit any $\perp$ block in M_1 without A_1's assistance. Note that this treatment has no side-effect in terms of security.

4.2 Security Analysis

Security of the above scheme is based on the security of the BLS scheme, which is proven in [5].

LEMMA 4.1. *If the BLS signature scheme is $(t, q_s, q_h, \varepsilon_1)$-unforgeable in the random oracle model, then our scheme is $(t', q'_s, q'_h, \varepsilon'_1)$-initiator-unforgeable in the random oracle model, where $t' = t - O((q_s + 1) \cdot \ell)T_{\mathsf{e}}$, $q'_s \ge \frac{q_s}{\ell} - 1$, $q_h = q'_h$, $\varepsilon_1 = \varepsilon'_1$ and T_{e} is the time cost of one exponentiation in $\mathbb{G}$.*

PROOF. We prove that if there exists an adversary $\mathcal{A}_{\mathsf{iu.cma}}$ who can $(t', q'_s, q'_h, \varepsilon'_1)$-break the initiator unforgeability of our scheme, then we can construct an adversary $\mathcal{A}_{\mathsf{BLS}}$ that can $(t, q_s, q_h, \varepsilon_1)$-break the BLS signature scheme, where $t = t' + O((q'_s + 1) \cdot \ell)T_{\mathsf{e}}$, $q_s \le (q'_s + 1) \cdot \ell$, $q_h = q'_h$ and $\varepsilon_1 = \varepsilon'_1$.

Suppose adversary $\mathcal{A}_{\mathsf{BLS}}$ obtains from its BLS-signature environment denoted by $\mathcal{E}_{\mathsf{BLS}}$ the system parameter $pp_{\mathsf{BLS}} = (\mathbb{G}, \mathbb{G}_T, p, g, e, H)$ and the challenge public key pk^* of the BLS signature scheme. The environment $\mathcal{E}_{\mathsf{BLS}}$ provides oracle $\mathsf{BLS.Sign}$ that returns BLS signatures on requested messages. Then adversary $\mathcal{A}_{\mathsf{BLS}}$ starts to run adversary $\mathcal{A}_{\mathsf{iu.cma}}$ with public parameter $pp = pp_{\mathsf{BLS}}$ where the challenge public key $pk^* = pk_1$.

For the oracle queries made by the adversary $\mathcal{A}_{\mathsf{iu.cma}}$, $\mathcal{A}_{\mathsf{BLS}}$ operates as follows:

- If the adversary $\mathcal{A}_{\mathsf{iu.cma}}$ wants to register pk_j, it must submit its corresponding private key sk_j as well as pk_j to $\mathcal{A}_{\mathsf{BLS}}$. (pk_j, c_{pk_j}, sk_j) is inserted into list C.

- For the FSign query on M, $\mathcal{A}_{\mathsf{BLS}}$ does the following:
  1. Parse $M = (m_1, \ldots, m_\ell)$.
  2. If $m_i \neq \perp$, query for the signature σ_i from oracle BLS.Sig on messages $(i||m_i||pk_1)$; otherwise let $\sigma_i = 1$.
  3. Compute
$$\sigma = \prod_{i=1}^{\ell} \sigma_i$$
  4. Pick up $\Delta \subseteq S$ and let
$$K_\Delta = \{s_i = \sigma_i | i \in \Delta\}.$$
  5. Set $\mathsf{edinfo} = (K_\Delta, \Delta)$ and $L = [(pk_1, \Gamma_1)]$ where $\Gamma_1 = S$.
  6. Output $(M, \sigma, \mathsf{edinfo}, L)$.
- If $\mathcal{A}_{\mathsf{iu.cma}}$ makes a query on H, $\mathcal{A}_{\mathsf{BLS}}$ responds with the answer from its own random oracle H.

Adversary $\mathcal{A}_{\mathsf{iu.cma}}$ finally outputs $M = (m_1, \ldots, m_\ell), \sigma, L = [(pk_1, \Gamma_1), \ldots, (pk_n, \Gamma_n)]$ to win the experiment $\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{iu.cma}}$. $\mathcal{A}_{\mathsf{BLS}}$ does the following to output its own forgery.

1. Find the corresponding private key sk_j of pk_j $(2 \leq j \leq n)$ in the list C and computes
$$\tilde{\sigma} = \sigma \cdot \prod_{2 \leq j \leq n} \left(\prod_{\alpha \in \Gamma_j \wedge m_\alpha \neq \perp} H(\alpha||m_\alpha||pk_j)^{-x_j} \right),$$
2. Let m_{α_0} be a message block s. t.
$$m_{\alpha_0} \notin \mathcal{M} \wedge \alpha_0 \in \Gamma_1 \wedge m_{\alpha_0} \neq \perp;$$
3. Query oracle BLS.Sig for signatures $\{\sigma_\alpha\}$ on messages $\{\alpha||m_\alpha||pk_1\}$ for α, s.t. $(\alpha \in \Gamma_1) \wedge (\alpha \neq \alpha_0) \wedge (m_\alpha \neq \perp) = 1$ (if not queried previously), output
$$\sigma_{\mathsf{BLS}} = \tilde{\sigma} \cdot \left(\prod_{\alpha \in \Gamma_1 \wedge \alpha \neq \alpha_0 \wedge m_\alpha \neq \perp} \sigma_\alpha \right)^{-1}.$$

It is easy to verify that σ_{BLS} is a valid BLS signature on $\alpha_0||m_{\alpha_0}||pk_1$ under pk^*. For the time cost of reduction, we have $t = t' + O(q)T_{\mathsf{e}}$, $q \leq (q'_s + 1) \cdot \ell$, $q_h = q'_h$ and $\varepsilon_1 = \varepsilon'_1$. □

LEMMA 4.2. *If the BLS signature scheme is $(t, q_s \cdot \ell, q_h, \varepsilon_2)$-unforgeable in the random oracle model, then our scheme is $(t', q'_s, q'_h, \varepsilon'_2)$-editor-unforgeable in the random oracle model, where $t = t' + O(q_s)\mathsf{p}$, $q_s \approx q'_s \cdot \ell$, $q_h = q'_h$, $\varepsilon_2 = \varepsilon'_2$ and p is the time cost of one pairing.*

PROOF. We prove that if there exists an adversary $\mathcal{A}_{\mathsf{eu.cma}}$ that can $(t', q'_s, q'_h, \varepsilon'_2)$-break the editor unforgeability of our scheme, then we can construct an adversary $\mathcal{A}_{\mathsf{BLS}}$ that can $(t, q_s, q_h, \varepsilon_2)$-break the BLS signature scheme, where $t = t' + O(q_s)\mathsf{p}$, $q_s \leq q'_s \cdot \ell$, $q_h = q'_h$ and $\varepsilon_2 = \varepsilon'_2$.

At first the adversary $\mathcal{A}_{\mathsf{BLS}}$ obtains from its environment $\mathcal{E}_{\mathsf{BLS}}$ the system parameter $pp_{\mathsf{BLS}} = (\mathbb{G}, \mathbb{G}_T, p, g, e, H)$ and challenge public key pk^* of the BLS signature scheme. The environment $\mathcal{E}_{\mathsf{BLS}}$ provides oracle BLS.Sig that returns the BLS signature on requested message, similar to the proof for INITIATOR UNFORGEABILITY. Then adversary $\mathcal{A}_{\mathsf{BLS}}$ starts to run adversary $\mathcal{A}_{\mathsf{eu.cma}}$ with public parameter $pp = pp_{\mathsf{BLS}}$ where the challenge public key $pk^* = pk_i$.

For the oracle queries made by the adversary $\mathcal{A}_{\mathsf{eu.cma}}$, $\mathcal{A}_{\mathsf{BLS}}$ does as follows:

- If the adversary $\mathcal{A}_{\mathsf{eu.cma}}$ wants to register pk_j for $j \neq i$, it must submit its corresponding private key sk_j as well as pk_j to $\mathcal{A}_{\mathsf{BLS}}$. (pk_j, c_{pk_j}, sk_j) is inserted into list certList.
- If $\mathcal{A}_{\mathsf{eu.cma}}$ makes a query on H, $\mathcal{A}_{\mathsf{BLS}}$ responds with the answer from its own random oracle H.
- For ESign query on $(M', M, \sigma, \mathsf{edinfo}, L)$, Parse $M = (m_1, \ldots, m_\ell)$, $L = [(pk_1, \Gamma_1), \ldots, (pk_{i-1}, \Gamma_{i-1})]$, $M' = (m'_1, \ldots, m'_\ell)$ and $\mathsf{edinfo} = (K_\Delta, \Delta)$, where $K_\Delta = \{s_\alpha | \alpha \in \Delta\}$ and $\bigcup_{j=1}^{i-1} \Gamma_j = S$.

  Let
$$\mathsf{H}(\alpha||m_\alpha||pk_j) = \begin{cases} 1, & \text{if } m_\alpha = \perp \\ H(\alpha||m_\alpha||pk_i), & \text{if } m_\alpha \neq \perp \end{cases}$$
  for all $1 \leq j \leq n$ and $\alpha \in S$.

  If
$$e(\sigma, g) \neq \prod_{j=1}^{i-1} e\left(\prod_{\beta \in \Gamma_j} \mathsf{H}(\beta||m_\beta||pk_j), pk_j \right),$$
  abort; else execute as follows:

  – In the case $1 < i < n$, execute the following:
    1. Let $C = \{\alpha | m'_\alpha = m_\alpha, \alpha \in S\}$
    2. Choose $I \subseteq C$ where I is the set of indices of messages blocks that A_i will sign. Let $\overline{I} = \Delta \setminus C$, which is the set of indices of message blocks to be edited by A_i.
    3. Query its own oracle BLS.Sig for
$$S_\alpha = H(\alpha||m'_\alpha||pk_i)^{x_i}$$
    where $\alpha \in (I \cup \overline{I}) \wedge m'_\alpha \neq \perp$.
    4. Define
$$\sigma_\alpha = \begin{cases} S_\alpha, & \text{for } m'_\alpha \neq \perp; \\ 1, & \text{for } m'_\alpha = \perp. \end{cases}$$
    5. Compute
$$\sigma' = \sigma \cdot \left(\prod_{\alpha \in I \cup \overline{I}} \sigma_\alpha \right) \cdot \left(\prod_{\alpha \in \overline{I}} s_\alpha^{-1} \right),$$
    6. Update $\Gamma_j = \Gamma_j \setminus (\overline{I})$ for $j = 1, \ldots, i-1$ and set $\Gamma_i = I \cup \overline{I}$;
    7. Pick up $\Delta' \subset \Delta$ and update
$$s'_\alpha = \begin{cases} \sigma_\alpha, & \text{for } \alpha \in \overline{I} \bigcap \Delta' \\ s_\alpha \cdot \sigma_\alpha, & \text{for } \alpha \in I \bigcap \Delta' \\ s_\alpha, & \text{for } \alpha \in (C \setminus I) \bigcap \Delta' \end{cases}$$
    and let
$$K_{\Delta'} = \{s'_\alpha | \alpha \in \Delta'\}.$$
    8. Set $L' = [(pk_1, \Gamma_1), \ldots, (pk_i, \Gamma_i)]$ and return
$$(M', \sigma', \mathsf{edinfo}', L')$$
    where $\mathsf{edinfo}' = (K_{\Delta'}, \Delta')$.

– In the case $i = n$, namely that the signer is the last one, the signer does the same as A_j ($j \neq n$) does, except that it sets $\mathsf{edinfo}' = \mathtt{null}$. The final signature output is (M', σ', L').

Adversary $\mathcal{A}_{\mathsf{eu.cma}}$ finally outputs $M = (m_1, \ldots, m_\ell), \sigma, L = [(pk_1, \Gamma_1), \ldots, (pk_n, \Gamma_n)]$ to win the experiment $\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{eu.cma}}$. $\mathcal{A}_{\mathsf{BLS}}$ does the following to output its own forgery.

1. Find the corresponding private key sk_j of pk_j ($1 \leq j \neq i \leq n$) in the list $\mathsf{certList}$ and computes
$$\tilde{\sigma} = \sigma \cdot \prod_{1 \leq j \neq i \leq n} \left(\prod_{\alpha \in \Gamma_j} \mathsf{H}(\alpha || m_\alpha || pk_j)^{-x_j} \right).$$
2. Let m_{α_0} be a message s. t.
$$m_{\alpha_0} \notin \mathcal{M} \wedge \alpha_0 \in \Gamma_i \wedge m_{\alpha_0} \neq \perp.$$
3. Query oracle $\mathsf{BLS.Sig}$ for signatures $\{\sigma_\alpha\}$ on messages $\{(\alpha || m_\alpha || pk_i)\}$ where
$$\Big[(\alpha \in \Gamma_i) \bigwedge (\alpha \neq \alpha_0) \bigwedge (m_\alpha \neq \perp) \Big] = \mathtt{true},$$
output
$$\sigma_{\mathsf{BLS}} = \tilde{\sigma} \cdot \left(\prod_{\alpha \in \Gamma_i \wedge \alpha \neq \alpha_0} \sigma_\alpha \right)^{-1}.$$

Therefore, it is easy to verify that σ_{BLS} is a valid BLS signature on $(\alpha_0 || m_{\alpha_0} || pk_i)$ under pk^*. For the time of reduction, we have $t = t' + O(q_s)\mathsf{p}$, $q_s \leq q'_s \cdot \ell$, $q_h = q'_h$ and $\varepsilon_2 = \varepsilon'_2$. □

THEOREM 4.3. *Our scheme is secure in the random oracle model if the CDH problem is hard on bilinear groups.*

PROOF. Because it was proven in [5] that the BLS scheme is unforgeable in the random oracle model based on the CDH assumption, the above two lemmas show that our scheme achieves both INITIATOR UNFORGEABILITY and EDITOR UNFORGEABILITY in the random oracle model based on the CDH assumption as well. □

Note that in the above scheme, the final editable signature of a message is just one group element of $\mathbb{G}$ (e.g., 160 bits if we use the parameters in [5]). However, the intermediate signatures should include the "editing keys" for the next editor, which could lead to $O(\ell)$ communication complexity. Verification also costs a linear number of pairings for a single signature. In summary, the resulting editable signatures are short, but their verification requires $\mathcal{O}(n)$ pairing operations. Next we will show a more efficient scheme that reduces the intermediate signature size and verification cost, but at the price of restricting the freedom of editors in terms of which message blocks they can edit.

5. NON-SWITCHABLE EDITABLE SIGNATURES

Now we present an editable signature scheme that allows the initiator (the first signer) assigns the blocks that can be edited by the respective editors, who however do not have such power. We call such a variant non-switchable editable signatures. This scheme is based on the Waters signatures [21] and only requires a constant number of pairing operations in verifying a signature. Its security is proven in the standard model (i.e., without using random oracle). Furthermore, this scheme doesn't require private channels between the signers (because the information does not need to be kept secret).

5.1 Construction

Let $\mathbb{G}$, $\mathbb{G}_T$, p, g, e be the bilinear parameters reviewed above (and as in [5, 21]). In this scheme, the initiator A_1 determines that the i-th editor can at most edit the i-th block m_i, meaning that $\ell = n$ in this case. Moreover, if A_i does modify m_i provided by A_1 to obtain m'_i, then A_i must sign m'_i; if A_i does not modify m_i provided by A_1, then A_i must sign m_i as well (compared with the previous scheme where A_i does not have to sign m_i, here A_i must either sign m_i or modify m_i).[2] The scheme does not require private channels between the signers because the "editing keys" are somehow embedded into the private keys of the signers. As such, there is no need for the auxiliary information edinfo. Moreover, Γ_i is not needed because it is regulated that the i-th block m_i must be signed by A_i (and by A_1 when A_i does not modify m_i). Therefore, L only needs to indicate who has already processed and signed the message.

The scheme $\mathsf{ES} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{FSign}, \mathsf{ESign}, \mathsf{Vf})$ is described as follows:

$\underline{\mathsf{Setup}(1^\lambda)}$**:** On input a security parameter λ, it outputs
$$pp = (\mathbb{G}, \mathbb{G}_T, p, g, e).$$

$\underline{\mathsf{Gen}(pp)}$**:** It takes pp as input, randomly chooses
$$(x_i, y_{i0}, y_{i1}, \ldots, y_{ik}) \xleftarrow{R} \mathbb{Z}_p^{(k+2)},$$
computes $\mu_i = (u_{i0}, u_{i1}, \ldots, u_{ik})$ where $u_{ij} = g^{y_{ij}}$ for $j = 0, 1, \ldots, k$ and $B_i = e(d, g^{x_i})$, outputs the i-th signer's private and public key
$$sk_i = (d^{x_i}, y_{i0}, \ldots, y_{ik}), \quad pk_i = (\mu_i, B_i).$$
Note that μ_i defines $H_{\mu_i} : \{0,1\}^k \longrightarrow \mathbb{G}$ by setting
$$H_{\mu_i}(m) = u_{i0} \left(\prod_{j=1}^{k} u_{ij}^{m[j]} \right),$$
where $m[j]$ is the j-th bit of message m, which itself could be a hash value, say $m = h(a)$ of some message a where $h : \{0,1\}^* \to \{0,1\}^k$. We can define
$$H(M) = \prod_{j=1}^{\ell} H_{\mu_i}(m_i)$$
where $M = (m_1, \ldots, m_\ell)$.

$\underline{\mathsf{FSign}(pp, sk_1, M)}$**:** It takes $M = (m_1, m_2, \ldots, m_\ell)$ as input, picks up $L = [\phi, (pk_1, \ldots, pk_\ell)]$ and computes the following:

1. Randomly choose $r \xleftarrow{R} \mathbb{Z}_p$ and compute
$$s = d^{x_1} \cdot \left(\prod_{i=1}^{\ell} H_{\mu_i}(m_i) \right)^r, \quad t = g^r,$$

[2] In a future work, we will show that it is possible to eliminate this restriction. Nevertheless, the security proof will become more involved.

and set $\sigma' = (s, t)$ and $M' = M$.

2. Set $L' = [(pk_1), (pk_2, \ldots, pk_\ell)]$ meaning that A_1 has already signed M, but the others haven't, output (M', σ', L').

$\underline{\mathsf{ESign}(pp, sk_i, M', M, \sigma, L)}$ **for** $1 < i \leq n$**:** Parse input message $M = (m_1, \ldots, m_\ell)$, input signature $\sigma = (s, t)$, list $L = [(pk_1, \ldots, pk_{i-1}), (pk_i, pk_{i+1}, \ldots, pk_\ell)]$ meaning that $A_1, \ldots, A_{i-1}$ have processed. Abort the execution if

$$e(s, g) \neq \prod_{j=1}^{i-1} B_j \cdot e\left(\prod_{j=1}^{\ell} H_{\mu_j}(m_j), t\right);$$

otherwise execute as follows:

- In the case $1 < i < n$, execute the following:
  1. Randomly choose r' from $\mathbb{Z}_p$ and compute $t' = t \cdot g^{r'}$.
  2. Parse $M' = (m_1, \ldots, m_{i-1}, m'_i, m_{i+1}, \ldots, m_\ell)$.
  3. If $m_i \neq m'_i$ (replacing m_i in M with m'_i in M'), compute

$$s' = d^{x_i} \cdot s \cdot t^{\left[\sum_{j=1}^{k} y_{ij}(m'_i[j] - m_i[j])\right]} \cdot \left(H(M')\right)^{r'},$$

  where $m_i[j]$ and $m'_i[j]$ are the j-th bits of message m_i and m'_i, respectively. Otherwise, ($m'_i = m_i$ and $M' = M$), compute $s' = d^{x_i} \cdot s \cdot H(M)^{r'}$.
  4. Let $L' = [(pk_1, \ldots, pk_i), (pk_{i+1}, \ldots, pk_\ell)]$, $\sigma' = (s', t')$ and send (M', σ', L') to the next signer.
- In the case $i = n$, execute the same as in the above except that (M', σ', L') is the final message and signature that can be released into an information network.

$\underline{\mathsf{Vf}(pp, M, \sigma, L)}$**:** Given parameters pp, $L = [(pk_1, \ldots, pk_\ell), \phi]$, a message $M = (m_1, \ldots, m_\ell)$ and an alleged signature $\sigma = (s, t)$, the verifier accepts the signature if

$$e(s, g) = e(H(M), t) \prod_{i=1}^{\ell} B_i, \tag{5.1}$$

and reject otherwise.

In order to further clarify the difference between the two schemes, we here reiterate that in the final signature, the first block m_1 is always signed by the initiator A_1 alone (i.e., not edited by any A_i with $i > 1$). Whereas, the i-th block m_i is first "drafted and signed" by A_1 and then processed by A_i in one of the following fashions: (i) A_i concurs with A_1 and thus signs m_i as well. In this case m_i is effectively signed with a multisignature by A_1 and A_i. (ii) A_i modifies m_i, which was provided by A_1, to obtain $m'_i \neq m_i$. In this case m'_i in the final message is only signed by A_i.

The non-switchable editable signature scheme is much more efficient than the fully editable signature scheme, in terms of both communication and computation since we do not require the initiator to transmit the auxiliary information. However, in addition to that the public keys are longer, a signature consists of two group elements that are longer than a signature in the previous scheme.

5.2 Security Results

LEMMA 5.1. *If the Waters signature scheme is (t, q, ε_1)-unforgeable, then our scheme is $(t - O(q), q, \varepsilon_1)$-initiator-unforgeable.*

PROOF. We prove that if there exists an adversary $\mathcal{A}$ that can (t', q', ε'_1)-break the initiator unforgeability of our non-switchable editable signature scheme, then we can construct an adversary $\mathcal{B}$ that can (t, q, ε_1)-break the Waters signature scheme, where $t = t' - O(q)$, $q' = q$ and $\varepsilon'_1 = \varepsilon_1$.

At the beginning we assume that adversary $\mathcal{B}$ obtains from its Waters signature environment denoted by $\mathcal{E}_{\mathsf{waters}}$ the system parameter $pp_{\mathsf{waters}} = (\mathbb{G}, \mathbb{G}_T, p, g, e, u_{10}, \{u_{1j}\}_{1 \leq j \leq k})$, and the challenge public key $pk_1 = pk^*_w$ of the Waters signature scheme. Here we assume that the message length of the underlying Waters signature scheme is k.

The environment $\mathcal{E}_{\mathsf{waters}}$ provides an oracle $\mathsf{W.Sig}$ that returns the Waters signature on requested message. Then adversary $\mathcal{B}$ starts adversary $\mathcal{A}$ with public parameter $pp = pp_{\mathsf{waters}}$ and the challenge public key $pk_1 = pk^* = pk^*_w$.

For the oracle queries made by the adversary $\mathcal{A}$, $\mathcal{B}$ executes as follows:

- If the adversary $\mathcal{A}$ wants to register $pk_v = (\mu_v, B_v)$ for $2 \leq v \leq \ell$ where $\mu_v = (u_{v0}, u_{v1}, \ldots, u_{vk})$ and $B_v = e(d, g^{x_v})$, it must submit its corresponding private key $sk_v = (d^{x_v}, y_{v0}, \ldots, y_{vk})$ where $u_{vj} = g^{y_{vj}}$ for $j = 0, 1, \ldots, k$, as well as pk_v to $\mathcal{B}$. Then (pk_v, c_{pk_v}, sk_v) is inserted into list $\mathsf{certList}$.
- For FSign query (M, L), $\mathcal{B}$ first parse $\sigma = (s, t)$, $L = [\phi, F]$ then does as follows:
  1. Parse $M = (m_1, m_2, \ldots, m_\ell)$;
  2. Make a $\mathsf{W.Sig}$ query from $\mathcal{E}_{\mathsf{waters}}$ and obtains the signature $\sigma_1 = (s_1, t_1)$ on message m_1 (if not queried) where
$$s_1 = d^{x_1} \cdot H_{\mu_1}(m_1)^r,$$
$$t_1 = g^r.$$
  3. Look for $sk_v = (d^{x_v}, y_{v0}, \ldots, y_{vk})$ from list $\mathsf{certList}$ for all $v \neq i$ and compute
$$e = \sum_{v=2}^{\ell} y_{v0} + \sum_{v=2}^{\ell} \sum_{j=1}^{k} (m_v[j] \cdot y_{vj}),$$
$$s' = s_1 \cdot t_1^e \cdot H(M')^{r'},$$
$$t' = t_1 \cdot g^{r'}.$$
  where $m_v[j]$ is the j-th bit of the v-th block of message m_v and r' is a random number in $\mathbb{Z}_p$.
  4. Let $E = (pk_1)$ and $F = (pk_2, \ldots, pk_\ell)$.
  5. Set $\sigma' = (s', t')$, $L' = [E, F]$, and output $(M' = M, \sigma', L')$ as the response of $\mathcal{B}$ to such a query.

Adversary $\mathcal{A}$ finally outputs $(M, \sigma = (s, t), L = [E, F])$ to win the experiment $\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{iu.cma}}$, where $F = \phi$. $\mathcal{B}$ parse $M = (m_1, \ldots, m_\ell)$, where $m_1 \notin \mathcal{M}$ since it is a valid forgery. Then, it executes the following to obtain a forgery of the Waters signature σ_w.

1. Look for $sk_v = (d^{x_v}, y_{v0}, \ldots, y_{vk})$ from list $\mathsf{certList}$ for all $v \neq 1$.

2. Compute

$$e = \sum_{v=2}^{\ell} y_{v0} + \sum_{v=2}^{\ell}\sum_{j=1}^{k}(m_v[j]\cdot y_{vj}),$$
$$s' = s\cdot t^{-e}\cdot\left(\prod_{v=2}^{\ell} d^{x_v}\right)^{-1},$$
$$t' = t.$$

where $m_v[j]$ is the j-th bit of the v-th block of message m_v.

3. Set $\sigma_w = (s', t')$.

Then, σ_w is a valid Waters signature on m_1. For time cost in reduction, we have $t = t' + O(q)$, $q' = q$ and $\varepsilon_1' = \varepsilon_1$. □

LEMMA 5.2. *If the Waters signature scheme is (t, q, ε_2)-unforgeable, then the above non-switchable editable signature scheme is $(t - O(q), q, \varepsilon_2)$-editor-unforgeable.*

PROOF. We prove that if there exists an adversary $\mathcal{A}$ that can (t', q', ε_2')-break the editor unforgeability of our non-switchable editable signature scheme, then we can construct an adversary $\mathcal{B}$ that can (t, q, ε_2)-break the Waters signature scheme, where $t' = t - O(q)$, $q' = q$ and $\varepsilon_2' = \varepsilon_2$.

At the beginning we assume that adversary $\mathcal{B}$ obtains from its environment $\mathcal{E}_{\mathsf{waters}}$ the system parameter $pp_{\mathsf{waters}} = (\mathbb{G}, \mathbb{G}_T, p, g, e, u_{i0}, \{u_{ij}\}_{1\le j\le k})$ for some fixed i, and challenge public key pk_w^* of the Waters signature scheme. Here we assume that the message length of the underlying Waters signature scheme is k.

The environment $\mathcal{E}_{\mathsf{waters}}$ provides an oracle $\mathsf{W.Sig}$ that returns the Waters signature on requested message. Then adversary $\mathcal{B}$ starts adversary $\mathcal{A}$ with public parameter $pp = pp_{\mathsf{waters}}$ and the challenge public key $pk_i = pk^* = pk_w^*$.

For the oracle queries made by the adversary $\mathcal{A}$, $\mathcal{B}$ executes the following:

- If the adversary $\mathcal{A}$ wants to register $pk_v = (\mu_v, B_v)$ for $1 \le v \ne i \le \ell$ where $\mu_v = (u_{v0}, u_{v1}, \ldots, u_{vk})$ and $B_v = e(d, g^{x_v})$, it must submit its corresponding private key $sk_v = (d^{x_v}, y_{v0}, \ldots, y_{vk})$ where $u_{vj} = g^{y_{vj}}$ for $j = 0, 1, \ldots, k$, as well as pk_v to $\mathcal{B}$. Then (pk_v, c_{pk_v}, sk_v) is inserted into list C.
- For ESign query (m_i', M, σ, L), $\mathcal{B}$ first parse $\sigma = (s, t)$, $L = [E, F]$ and check the validity by

$$e(s, g) \stackrel{?}{=} \prod_{pk_j \in E} B_j \cdot e\left(t, H(M)\right),$$

if invalid, abort; otherwise

1. Make a $\mathsf{W.Sig}$ query from $\mathcal{E}_{\mathsf{waters}}$ and obtains the signature (s_i, t_i) on message m_i' where

$$s_i = d^{x_i}\cdot H_{\mu_i}(m_i')^r,$$
$$t_i = g^r.$$

2. Let $M' = (m_1, m_2, \ldots, m_i', m_{i+1}, \ldots, m_\ell)$ (replacing m_i by m_i' in M) and compute

$$e = \sum_{v\ne i} y_{v0} + \sum_{v\ne i}\sum_{j=1}^{k}(m_v[j]\cdot y_{vj}),$$
$$s' = s_i\cdot t_i^e\cdot H(M')^{r'},$$
$$t' = t_i\cdot g^{r'}.$$

where $m_v[j]$ is the j-th bit of the v-th block of message m_v and r' is a random number in $\mathbb{Z}_p$.

3. Let $E = (pk_1, \ldots, pk_i)$, $F = (pk_{i+1}, \ldots, pk_\ell)$
4. Set $\sigma' = (s', t')$, $L' = [E, F]$, and output (M', σ', L') as the response of $\mathcal{B}$ to such a query.

Adversary $\mathcal{A}$ finally outputs $(M, \sigma = (s, t), L = [E, F])$ to win the experiment $\mathrm{Exp}^{\mathsf{ES}}_{\mathsf{eu.cma}}$, where $F = \phi$. $\mathcal{B}$ parse $M = (m_1, \ldots, m_\ell)$, where $m_i \notin \mathcal{M}$ since it is a valid forgery. Then, does the following to obtain a forgery of the Waters signature σ_w

1. Look for $sk_v = (d^{x_v}, y_{v0}, \ldots, y_{vk})$ from list $\mathsf{certList}$ for all $v \ne i$.
2. Compute

$$e = \sum_{v\ne i} y_{v0} + \sum_{v\ne i}\sum_{j=1}^{k}(m_v[j]\cdot y_{vj}),$$
$$s' = s\cdot t^{-e}\cdot\left(\prod_{v\ne i} d^{x_v}\right)^{-1},$$
$$t' = t.$$

where $m_v[j]$ is the j-th bit of the v-th block of message m_v.

3. Set $\sigma_w = (s', t')$.

Then, σ_w is a valid Waters signature on m_i. For time cost in reduction, we have $t' = t - O(q)$, $q' = q$ and $\varepsilon_2' = \varepsilon_2$. □

THEOREM 5.3. *Our scheme is secure if the CDH problem in bilinear groups is hard.*

PROOF. Since the Waters signature scheme is unforgeable under the CDH assumption, the above two lemmas show that our scheme is INITIATOR-UNFORGEABLE and EDITOR-UNFORGEABLE without using random oracles based on the CDH assumption. □

6. RELATED WORK

While there have been many studies on data provenance [6, 19, 20, 11, 10, 3], the security aspect of data provenance has not been investigated until very recently [7, 12, 24, 23, 14, 22]. However, existing studies focused on what we called passive data provenance. In this paper, we introduced the novel concept of proactive or assured data provenance, which further guided us to propose the novel cryptographic technique we call editable signatures, which are a building-block for achieving assured data provenance management. Coincidentally, as shown in Section 2, the concept of editable signatures generalizes both multisignatures and aggregate signatures.

7. CONCLUSION

We introduced the novel concept of assured data provenance which, unlike existing passive data provenance approach, aims to "distill" less trustworthy data before they enter into an information network. This concept further guided us to propose a novel cryptographic tool, called editable signatures, which can be adopt to facilitate assured provenance management.

There are many questions for future research. In addition to those mentioned in the body of the paper, here we give more examples. On one hand, it would be very interesting to explore a full-fledged framework for characterizing the concept of assured data provenance. On the other hand, it is interesting to construct full-fledged editable signature schemes that operate in the weakest plain public-key (PPK) model [2] (where so-called "rogue key attack" is allowed) without random oracles. Our first scheme was proven secure in the KOSK model, but we believe that it can be proven secure in the weakest PPK model. However, this scheme is not as efficient as our second scheme, which is nevertheless difficult to be made secure in the PPK model. It is also interesting to construct practical editable signature models in the weaker operational models [17] such as the Proof of Possession (POP) of private key model [18].

Acknowledgements.

Haifeng Qian is partially supported by the National Natural Science General Foundation of China Grant No. 60703004, 60873217 and 61021004, the Research Fund for the Doctoral Program of Higher Education of China Grant No. 20070269005. Shouhuai Xu is supported in part by an AFOSR MURI grant and a State of Texas Emerging Technology Fund grant.

8. REFERENCES

[1] M. Bellare, C. Namprempre, and G. Neven. Unrestricted aggregate signatures. In *In ICALP, 2007.*, pages 9–13. Springer-Verlag, 2006.

[2] M. Bellare and G. Neven. Multisignatures in the plain public-key model and a general forking lemma. In *ACM Conference on Computer and Communications Security (CCS'06)*, pages 390–399, 2006.

[3] O. Benjelloun, A. Sarma, A. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures. In *EUROCRYPT'03*, pages 416–432, 2003.

[5] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Asiacrypt'01*, pages 514–532, 2002.

[6] R. Bose and J. Frew. Lineage retrieval for scientific data processing: a survey. *ACM Comput. Surv.*, 37(1):1–28, 2005.

[7] U. Braun, A. Shinnar, and M. Seltzer. Securing provenance. In *HotSec'08*, 2008.

[8] P. Buneman, S. Khanna, and W. Tan. Why and where: A characterization of data provenance. In *Proceedings of the 8th International Conference on Database Theory (ICDT'01)*, pages 316–330, 2001.

[9] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2).

[10] T. Green, G. Karvounarakis, Z. Ives, and V. Tannen. Update exchange with mappings and provenance. In *VLDB*, 2007.

[11] T. Green, G. Karvounarakis, N. Taylor, O. Biton, Z. Ives, and V. Tannen. Orchestra: facilitating collaborative data sharing. In *SIGMOD'07*, pages 1131–1133, 2007.

[12] R. Hasan, R. Sion, and M. Winslett. The case of the fake picasso: preventing history forgery with secure provenance. In *Proccedings of the 7th conference on File and storage technologies (FAST'09)*, pages 1–14, 2009.

[13] K. Itakura and K. Nakamura. A public key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71:1–8, 1983.

[14] J. Lyle and A. Martin. Trusted computing and provenance: Better together. In *Proceedings of 2nd USENIX Workshop on the Theory and Practice of Provenance (TaPP'10)*, 2010.

[15] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *EUROCRYPT*, pages 74–90, 2004.

[16] G. Neven. Efficient sequential aggregate signed data. In *EUROCRYPT*, pages 52–69, 2008.

[17] H. Qian and S. Xu. Non-interactive multisignatures in the plain public-key model with efficient verification. *Inf. Process. Lett.*, 111(2):82–89, 2010.

[18] T. Ristenpart and S. Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In *EUROCRYPT*, pages 228–245, 2007.

[19] W. Tan. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.*, 30(4):3–12, 2007.

[20] N. Taylor and Z. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *SIGMOD'06*, pages 13–24, 2006.

[21] B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT'05*, pages 114–127.

[22] S. Xu, H. Qian, F. Wang, Z. Zhan, E. Bertino, and R. Sandhu. Trustworthy information: Concepts and mechanisms. In *Proceedings of 11th International Conference Web-Age Information Management (WAIM'10)*, pages 398–404, 2010.

[23] S. Xu, R. Sandhu, and E. Bertino. Tiupam: A framework for trustworthiness-centric information sharing. In *Proc. 2009 IFIP Trust Management Conference (TM'09)*, 2009.

[24] J. Zhang, A. Chapman, and K. Lefevre. Do you know where your data's been? — tamper-evident database provenance. In *Proceedings of the 6th VLDB Workshop on Secure Data Management (SDM'09)*, pages 17–32, 2009.