Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems*

Siamak F Shahandashti [◇] Reihaneh Safavi-Naini [△]

[◇] School of Computer Science and Software Engineering
University of Wollongong, Australia
http://www.uow.edu.au/∼sfs166

[△] Department of Computer Science
University of Calgary, Canada
http://www.cpsc.ucalgary.ca/∼rei

April 2, 2009

Abstarct. Inspired by the recent developments in attribute-based encryption, in this paper we propose *threshold attribute-based signatures* (t-ABS). In a t-ABS, signers are associated with a set of attributes and verification of a signed document against a verification attribute set succeeds if the signer has a threshold number of (at least t) attributes in common with the verification attribute set. A t-ABS scheme enables a signature holder to prove possession of signatures by revealing only the relevant (to the verification attribute set) attributes of the signer, hence providing *signer-attribute privacy* for the signature holder. We define t-ABS schemes, formalize their security and propose two t-ABS schemes: a basic scheme that is selectively unforgeable and a second one that is existentially unforgeable, both provable in the standard model, assuming hardness of the computational Diffie-Hellman problem. We show that our basic t-ABS scheme can be augmented with two extra protocols that are used for efficiently issuing and verifying t-ABS signatures on committed values. We call the augmented scheme a threshold attribute based c-signature scheme (t-ABCS). We show how a t-ABCS scheme can be used to realize a secure *threshold attribute-based anonymous credential system* (t-ABACS) providing signer-attribute privacy. We propose a security model for t-ABACS and give a concrete scheme using t-ABCS scheme. Using the simulation paradigm, we prove that the credential system is secure if the t-ABCS scheme is secure.

Keywords: Public-Key Cryptography, Attribute-Based Cryptography, Anonymous Credential Systems, Identity-based Cryptography, User Privacy, Fuzzy Identity-based Signature

*An abridged version of this paper is to appear in AfricaCrypt 2009 proceedings. This is the full version.

Contents

1 Introduction

Inspired by the recent works on attribute based encryption, we introduce a new signature scheme that we call a *threshold attribute-based signature*, or t-ABS for short. In a t-ABS scheme, a central authority issues secret keys to the signers according to their *attributes*. A signer with an attribute set Att can use his secret key to sign using any subset $A \subset Att$ of attributes. A signature can be verified against a verification attribute set B and verification succeeds if $A \cap B$ has at least size d. Threshold attribute-based signatures have attractive applications that are outlined below.

As their name suggests, t-ABS schemes provide *threshold attribute-based verification*. Consider the following import policy for goods to a country: imports from certain 'low-risk countries' can be from any company that has a specific ISO or a particular FAO certificate, but imports from other countries must be from a company with both certificates. This policy can be enforced by verifying if the exporting company has two of the three following attributes: it is in a low-risk country, it is an ISO certificate holder, and it is a FAO certificate holder. Here, a t-ABS scheme with appropriate definition of attributes can be used to implement the verification functionality.

The threshold verification can also be used for applications such as identity-based signatures that use biometric information as user identities to provide strong authentication for users. Here, key generating authorities measure users' biometrics and issue secret keys to them. Verifiers, measure users' biometrics independently to obtain their identity in the system. The new measurement will not be exactly the same as the ones collected at the registration phase and hence *fuzzy verification* is required. Using a threshold attribute-based signature scheme with suitable choices of mapping and threshold, "close" biometric measurements map to attribute sets for signing and verification that have sufficient overlap and result in successful verification of signature.

A t-ABS scheme also enables the signer to choose their "signing identity" (relevant attributes) at the time of signing. A professor in the School of Computer Science who is also Head of the School, may need to sign documents as either a Professor of Computer Science, or the Head of the School, or a Professor of Computer Science and the Head of the School. In general a person with n "atomic" identities can sign using an identity that is any of the $\mathrm{O}(2^n)$ "combination" identities, assuming the combination is meaningful. This can be implemented using a t-ABS scheme with a signing key of size $\mathrm{O}(n)$. Providing the same functionality using standard signatures or identity-based signatures requires $\mathrm{O}(2^n)$ signing keys, one for each combined identity. More importantly, the naïve solution of $\mathrm{O}(n)$ signing keys of a standard or an identity-based signature, is not secure because signatures can be "mixed and matched" with each other. A group of signers, for example, can collude to produce a signature corresponding to the union of their identities. A t-ABS scheme provides *collusion resistance*: that is, no group of colluding signers can produce a signature that could not be generated by a single member of the group.

A t-ABS scheme also enables *signature holders* to present for verification a signature that corresponds to subset of signer's attributes. A document signed by John Smith, a member of 2008 Executive Committee of the Workers' Union (WU), can be presented as signed by an Executive Committee member of the WU, without revealing the signer's name or the year. The same document can also be presented as signed by a 2008 Executive Committee member of the WU, or simply by John Smith. This functionality gives the signature holder the flexibility to present the same signature in different ways at different occasions. To provide the same functionality, using a standard or an identity-based signature scheme, requires $\mathrm{O}(2^n)$ signatures, each produced using one of the aforementioned $\mathrm{O}(2^n)$ keys. Using a t-ABS scheme however, requires a signing key of size $\mathrm{O}(n)$ and a signature of size $\mathrm{O}(n)$ [1].

A t-ABS scheme supports *signer-attribute privacy* for users. Consider a scenario where participation in a poll requires proof of residency in one of the 27 counties in a state. Assume these proofs are cards that are signed by either the local government or police authorities of the counties and are given to the long-time residents and foreign workers, respectively. For privacy reasons, card holders may want to be able to prove

[1] We note that in certain cases it may not be a desirable to be able to break up the attribute set during signing a message or showing a signature, in which case traditional methods based on standard or identity-based signatures could be used.

that they own a card but protect their other details. Proving possession of such a card is equivalent to proving possession of a card issued by one of the 2×27 possible entities. Using standard or identity-based signatures, a voter can prove possession of a valid signature with respect to one of the 2×27 public keys or identities through a proof of size 2×27 (see *proofs of partial knowledge* [CDS94]). A t-ABS can provide a proof of size 2+27 and ensures that the attributes of the signer (and hence those of the card holder) are not revealed. Signers in such schemes will each have two attributes: one corresponding to 'police' or 'local government', and the other representing one of the 27 counties. Verification of residency is then equivalent to requiring that a card is verifiable with a threshold of $d = 2$ attributes from the set of all 2+27 attributes above. This is because no signer can simultaneously be both police and government, or from two (or more) counties.

The above example can be extended to the more general case where the universal set of attributes $\mathbb{U}$ can be partitioned into n subsets $\mathbb{U}_i$ for $i \in N = \{1, 2, \ldots, n\}$ such that each signer has at most one attribute from each set. Now suppose providing service to signature holder requires the signer to possess an i-th attribute in the set B_i, for $i \in I \subseteq N$. Verification of such a property is equivalent to verifying if the signer has at least $|I| = d$ of the attributes in the attribute set $\mathbb{U}$. A t-ABS scheme can be used to implement the above system, with a verification cost of $\mathrm{O}(\sum_{i \in I} |B_i|)$. However, the number of possible accepting issuers for such a verification policy and hence the cost of a proof of partial knowledge is $\mathrm{O}(\prod_{i \in I} |B_i| \cdot \prod_{i \in N \setminus I} |\mathbb{U}_i|)$.

The above functionalities of t-ABS schemes become more attractive in cases where it is practically infeasible for the signature holder and the verifier to know all possible signers. For example, when a user wants to show that their document is signed by a notary public. It is impossible in practice for a verifier to know all the notary publics and thus standard or identity-based signatures cannot be used. Using a t-ABS however, allows the verifier to easily check if the signer has the attribute `notary public`.

As a prime example to show their application range, we show that t-ABS schemes, when augmented with additional protocols for signing and proof of signature ownership, can be used to construct *attribute-based anonymous credential systems*. In Section 5 we show that such credential systems inherit the above properties of t-ABS and provide a number of attractive features including *attribute privacy*, which is an essential property in anonymous credential systems, since verifying credentials against specific public keys reveals unnecessary information about the users and contradicts the "raison d'être" of such systems.

1.1 Our Contributions

We first introduce a new type of signature called *threshold attribute-based signature*, or t-ABS for short. In a t-ABS, a signer has an attribute set *Att* and receives a signing key from a key generating authority based on the set *Att*. The signer can sign a document using any subset $A \subset Att$ of attributes. A signature is successfully verified against an attribute set B as long as the signing set A and the verification set B have an intersection of size at least d. In other words, the verification succeeds if the attribute set used in the signature is sufficiently 'close' to the attribute set used for verification. We construct a t-ABS scheme in Section 3 and prove its security against selective forgery in the standard model based on the CDH assumption. We discuss how security and efficiency of the scheme can be improved and give a secure construction against existential forgery in the standard model.

We then introduce an algorithm for *converting* a signature and an interactive protocol for signature verification. The convert algorithm enables the signature holder to construct a signature that is verifiable with the attribute set $A \cap B$ and prevents the verifier from learning the signer's attributes that are outside $A \cap B$. The interactive verification protocol enables the signature holder to prove possession of a valid signature without allowing the verifier to learn even $A \cap B$. The two protocols, depending on the requirements of the application, can be used to provide two levels of *signer-attribute privacy*. The weak privacy level is achieved by providing the verifier with the converted signature and guarantees that the verifier does not find out any signer attribute outside $A \cap B$. The full privacy level is achieved by first converting the signature and then proving possession of the signature using the interactive verification protocol. This guarantees that the verifier learns nothing more than the fact that $|A \cap B| \geq d$. We denote a t-ABS with an efficient conversion algorithm and an efficient interactive verification protocol by t-ABS$^+$. We provide efficient conversion and interactive verification for our proposed basic scheme and prove that it provides both levels of signer-attribute privacy. We also show how signatures in the second construction can be efficiently converted and interactively verified and similarly prove our second construction to be fully signer-attribute private.

A t-ABCS scheme consists of a t-ABS$^+$ scheme, a commitment scheme, and three additional protocols for (i) obtaining a signature on a committed value, (ii) proving knowledge of a signature, and (iii) proving knowledge of a signature on a committed value. We give security definitions for t-ABCS schemes and a concrete efficient construction that is based on our basic t-ABS scheme and prove security of the construction. Consider a user with a secret key SK_U that has established a pseudonym N_{UI} with an issuer I and another pseudonym N_{UV} with a verifier V. Assume that these pseudonyms are both commitments to SK_U, but under different randomnesses. A t-ABCS scheme can be used by U to, on the pseudonym N_{UI}, obtain a credential that is associated with a signing attribute set A_I, and later provide proof of possession for the credential on the pseudonym N_{UV} against a verification attribute set B_V if $|A_I \cap B_V| \geq d$. The verifier cannot deduce any information about the d attributes in common between the signing attribute set A_I and the verification attribute set B_V; hence the privacy of the attributes presented by the credential holder is guaranteed.

We define threshold attribute-based anonymous credential systems (t-ABACS) and formalize their security using the *simulation paradigm* that is commonly used for defining security of multiparty protocols. The approach uses a comparison of a real system model with an ideal system model. The ideal model for the t-ABACS captures the security and privacy properties of an anonymous credential system. We show how t-ABCS schemes can be used to realize this model. That is, we give a concrete t-ABACS system and prove its security in the sense that it remains indistinguishable from an ideal system from the viewpoint of any polynomial-time adversary. This results in a concrete t-ABCS scheme with security based on the CDH problem.

1.2 Related Work

Attribute-based encryption (ABE) was introduced by Sahai and Waters as an extension of identity-based encryption where each identity is considered as a set of descriptive attributes [SW05]. The scheme, called fuzzy identity-based encryption, allows a threshold attribute-based decryption of encrypted data as follows. Messages can be encrypted by specifying a set of decryptor attributes ω' during encryption. Such a ciphertext then can be decrypted by any user with the attribute set ω such that $|\omega \cap \omega'| \geq d$. Our attribute-based signature scheme can be viewed as the signature counterpart of Sahai and Waters's encryption scheme. In our scheme, a signer has a set of attributes A and the verifier specifies a verification attribute set B. A signature is verified as valid if $|A \cap B| \geq d$.

Attribute-based signatures extend the *identity-based signature* of Shamir [Sha84] by allowing identity of a signer to be a set of descriptive attributes rather than a single string representing the signer's identity. A t-ABS provides threshold attribute-based verification. That is, the verification succeeds if at least d of the attributes specified by the verifier are the same as those of the signer. Thus, identity-based signature can be seen as a specific case of our schemes with identity size and threshold both equal to one.

Independent of our work, there have been other attempts to define and realize *attribute-based signatures* [Kha07b, Kha07a, YCD08, Kha08, GZ08, MPR08, LK08]. The schemes of [YCD08] and [GZ08] are direct applications of the known transform from identity-based encryptions to identity-based signatures [GS02]. In both works, authors do not consider any notion of privacy. The works of [Kha07b, Kha07a] and [LK08] capture weaker notions of anonymity where signers only remain anonymous within the group of entities possessing the same attributes and the verifiers must know which attributes are used to sign a message to be able to verify. Khader [Kha08] and Maji et al. [MPR08] treat attribute-privacy as a fundamental requirement of attribute-based signatures and provide schemes that provably satisfy this requirement for threshold trees and monotone boolean verification policies, respectively. However, both works have shortcomings that make their solutions unsuitable for the scenarios outlined in the introduction. Both schemes require the signer to know the verification policy at the time of signing. This is a major limitation for their proposed scheme, particularly in applications where other than the signer and the verifier, there are also 'signature holders'. An example of such applications is a credential system where a credential holder needs to satisfy different verification policies depending on the occasion, and it is important for efficiency and useability purposes not to require different credentials for each verification policy. Also, security proofs of Khader and Maji et al. are in the random oracle or generic group models, while all our proofs are in the standard model and use the well-known assumption of computational Diffie-Hellman. Finally, none of the previous attribute-based signatures had been extended to credential systems.

Anonymous credential systems (a.k.a. *pseudonym systems*) were introduced by Chaum [Cha85, CE87] and more recently further formalized and studied in [LRSW99, Lys02]. A credential in such systems is issued and verified on a user pseudonym, which in turn is bound to the user's secret identity. Users remain anonymous since their pseudonyms hide their secret identity. Besides, transactions involving the same user remain unlinkable.

2 Notation and Preliminaries

We use different font families to denote *variables*, $\mathsf{algorithms}$, `strings`, and SECURITY NOTIONS, respectively. By "$x \leftarrow \mathsf{X}\,(a)$" we denote that X is run on input a and the output is assigned to x. We also use "$A \dashv\!(\,X\,)\!\!\rightarrow B$ if C" to denote that A sends X to B if condition C holds. The condition can be complex and include logical connectors. The symbol $\setminus$ denotes set subtraction operation and we use $|S|$ to denote cardinality of the set S. We use the proof of knowledge notation originated in [CS97]. For instance, "$\mathsf{ZK\text{-}PoK}\{(x, y, z) : a = g^x h^y \wedge b = c^y d^z\}$" denotes a zero knowledge proof of knowledge of (x, y, z) such that $a = g^x h^y$ and $b = c^y d^z$, where a, g, h, b, c , and d are public inputs to the protocol.

Lagrange interpolation for a polynomial $q(\cdot)$ over $\mathbb{Z}_p$ of order $d - 1$ and a set $S \subset \mathbb{Z}_p$ with size $|S| = d$ is calculated as $q(x) = \sum_{i \in S} q(i)\Delta_{i,S}(x)$, where

$$\Delta_{i,S}(x) \overset{\triangle}{=} \quad 1 \cdot \prod_{j \in S, j \neq i} \frac{x - j}{i - j} \quad \text{for all } i \in S \quad (\text{and extendedly for all } i \in \mathbb{Z}_p) \ .$$

A *zero knowledge* (ZK) proof is intuitively a proof that although convinces a verifier, but adds no knowledge (i.e., ability to compute) to the verifier. Formally, a proof is called zero knowledge if the view of any verifier can be simulated given only the public inputs of the protocol (see [GMR89] for details). A protocol is called a *proof of knowledge* (PoK) if, intuitively, it guarantees that a successful prover actually knows (i.e., is able to compute) the secret in question. More formally, a protocol is said to be a proof of knowledge if there exists an extractor that is able to output the secret given the public inputs and access to a successful prover (see [BG92] for details). We use zero knowledge proofs of knowledge for conjunctive statements about discrete logarithms. Efficient versions of such proofs can be found in the literature. e.g., in the work by Cramer et al. [CDM00].

A *commitment scheme* consists of two algorithms $\mathsf{CmtKeyGen}$ and Commit, where the former is the key generation algorithm that generates a commitment public key *cpk* on input a security parameter and the latter is the commitment algorithm that on input a commitment public key *cpk*, a message to be committed m, and a random element r generates a commitment C on m. The scheme is said to be *hiding* if C hides the message m, that is, given *cpk* and C, it is hard to obtain any information about m. The scheme is said to be *binding* if C binds the committer to m, that is, Given *cpk*, it is hard to come up with two different pairs (m, r) and (m', r') such that $\mathsf{Commit}(cpk, m, r) = \mathsf{Commit}(cpk, m', r')$. The scheme is said to be secure if it has both of the above properties.

Camenisch and Lysyanskaya have shown that a tuple consisting of a signature, a commitment scheme, and efficient protocols for issuing and verifying signatures on committed values, is sufficient for realizing anonymous credential schemes [CL01, Lys02]. We denote such tuples by *C-signatures*.

A well-known paradigm that is used to define security of cryptographic protocols is the simulation (a.k.a. "ideal vs. real") paradigm, originating from [GMW91]. The intuition behind such definitions is that a real system is secure if it emulates a certain ideal system designed to trivially guarantee security properties expected from the system. To formally define security using this paradigm, one first defines a protocol-independent *framework*, which contains the model of computation and communication and a notion of *emulation*. The former determines the types of entities involved in the real-life and ideal protocol execution, what each of them can see and do, and how they communicate. The latter defines what it means to say a real protocol execution emulates (i.e., is as secure as) the ideal protocol execution. Then, an *ideal model* specific to the protocol is defined to capture the expected security properties. Security is guaranteed by including a trusted entity that collects the inputs of different entities in each round, performs the necessary calculations locally, and hands each entity's outputs for that round back to them. We use this paradigm to define a framework for analyzing security of our credential system.

3 Threshold Attribute-Based Signatures

We assume there is a universal set of attributes $\mathbb{U}$ that is publicly known. Each signer is associated with a subset $Att \subset \mathbb{U}$ of attributes that is verified by a central authority. In the following, we assume that the signing attribute set A is equal to the signer attribute set Att and we use the terms interchangeably. This has minimal impact in our definitions of the schemes and their security. We define the signature for a fixed threshold d. We discuss later how flexible thresholds can be achieved.

3.1 Definition

A threshold attribute-based signature (t-ABS) is a quadruple of algorithms as follows:

Setup is the algorithm run by a central authority on input the security parameter k and outputs a master secret key msk and a master public key mpk.

KeyGen is the algorithm run by the central authority on inputs msk and a set of signer attributes A and generates a secret signing key ssk for the signer.

Sign is the algorithm run by a signer on inputs ssk and a message m and generates a signature σ on the message.

Verify is the algorithm run by a verifier on inputs mpk, a message signature pair (m, σ), and a verification attribute set B and outputs 1 if σ is a valid signature by a signer who has at least d of the attributes in B, i.e., if $|A \cap B| \geq d$.

Correctness: A t-ABS scheme has to satisfy the correctness property, i.e., a signature generated by a signer with attributes A must pass the verification test for any B if $|A \cap B| \geq d$.

Unforgeability: For a t-ABS scheme defined as above we require that it is *existentially unforgeable* against *chosen message and attribute set attacks*. In particular, we define the following game between a challenger and the adversary.

Setup Phase: The challenger runs the Setup algorithm and gives mpk to the adversary.

Query Phase: The adversary is allowed to ask queries for the following:

- a secret key of a signer with attributes of its choice α, and
- a signature of a signer with any attribute set of its choice α on a message of its choice m.

Forgery Phase: The adversary outputs a triplet (μ, σ, β), consisting of a message μ, a forged signature σ, and a verification attribute set β, and wins if σ is a valid signature with respect to (mpk, μ, β) and

- for all queried sets of attributes α, we have $|\alpha \cap \beta| < d$, and
- for all queried pairs (α, m), we have $m \neq \mu$ or $|\alpha \cap \beta| < d$.

If no polynomial adversary has a considerable advantage in the above game, we say that the t-ABS scheme is existentially unforgeable against chosen message and attribute set attacks, or EUF-CMAA-secure for short. We also consider a weaker notion of security, *selective unforgeability* against chosen message and attribute set attacks (SUF-CMAA-security) in which the adversary should commit to the target forgery message and verification attribute set in the beginning of the attack. We define this notion in Appendix A and discuss how an existentially unforgeable scheme can be constructed given a selectively unforgeable scheme.

Collusion Resistance: It is important to note that the above definition of unforgeability guarantees *collusion resistance* in the sense that no colluding group of users can generate a signature that is not generable by one of the colluders. This is because if a group of signers can construct a signature that none of them could individually produce, then this is a forgery as per the above definition. Note that the adversary is able to query for secret keys for entities with chosen attributes, hence the game captures the scenario in which multiple signers are colluding. Furthermore, for the adversary to win, we must have $|\alpha \cap \beta| < d$ for *all* queried sets of attributes α. This ensures that none of the colluders could generate the forged signature by himself.

3.2 Additional Protocols

A signature holder can always check a signature σ against possible verification attribute sets to deduce information about the signer's attributes. To preserve privacy of signers we equip our t-ABS scheme with an additional algorithm for *converting* the signature to another signature that is verifiable against B and only reveals the d chosen attributes of the signer. The *converted signature* can be seen as a B-designated signature that contains a minimal subset of attributes from the original set of signer attributes, that allows the verification to succeed. Attribute privacy is obtained by using an *interactive verification* protocol iVerify, that allows the signature holder to prove possession of a valid converted signature without revealing the chosen d attributes in common between A and B.

We call our t-ABS scheme equipped with both the above conversion algorithm and interactive verification protocol a t-ABS$^+$ scheme, which formally contains the Setup, KeyGen, and Sign algorithms as defined in the t-ABS scheme plus the following:

Convert is the algorithm run by a signature holder on inputs mpk, a message signature pair (m, σ), and a verification attribute set B and generates a converted signature $\tilde{\sigma}$ on the message.

CvtVerify is the algorithm run by a verifier on inputs mpk, a message converted-signature pair $(m, \tilde{\sigma})$, and a verification attribute set B and outputs 1 if $\tilde{\sigma}$ is a valid converted signature by a signer who has at least d of the attributes in B, i.e., if $|A \cap B| \geq d$.

iVerify is an interactive verification protocol for proving knowledge of a converted signature on the *prover* side and verifying a converted signature on the *verifier* side. The public inputs are the t-ABS master public key mpk, a message m, and verifier's verification attribute set B. Prover's private input is a converted signature $\tilde{\sigma}$. The verifier has no private input. At the end of the protocol execution the verifier will output a binary value reflecting prover's converted signature validity against B.

Correctness: Apart from correctness of the underlying t-ABS scheme, we require that any converted signature calculated from a valid signature (i) passes the CvtVerify test, and (ii) makes the verifier in the iVerify protocol accept.

Unforgeability: We require that converted signatures in our t-ABS$^+$ scheme are also existentially unforgeable under chosen message and attribute set attacks. Since knowledge of a signature is sufficient for producing a converted signature (using algorithm Convert), converted signature unforgeability implies signature unforgeability, and hence, is a stronger notion of security. Converted signature existential unforgeability under chosen message and attribute set attacks (C-EUF-CMAA-security) is defined through a game, in which the setup and query phases are the same as the EUF-CMAA-security game above, but σ in the forgery phase is replaced with $\tilde{\sigma}$. That is, the adversary is given the same resources, but is expected to forge a nontrivial valid *converted* signature instead of a nontrivial valid signature. The full game is transcribed in Appendix A for completion.

Weak Signer-Attribute Privacy: A converted signature should not reveal any attribute of the signer other than the d of them common with B chosen by the signature holder at the time of conversion. Thus, we require that whatever a verifier can deduce about other attributes of the signer given a converted signature, can also be deduced given merely the d attributes as well. This ensures that only the d attributes of the signer that are chosen by the signature holder are revealed to the verifier given a converted signature. We call this property *weak signer-attribute privacy.*

Signer-Attribute Privacy: We also require that the iVerify protocol is a zero knowledge proof of knowledge of a valid converted signature with respect to the public inputs (mpk, m, β). This ensures that (i) only provers in possession of a valid converted signature are indeed successful in proving so, and (ii) the proof reveals no information other than the validity of the prover's converted signature to the verifier. We call this property *(full) signer-attribute privacy.* Note that property (ii) guarantees that proofs of possession of signatures from different signers satisfying the verification policy remain indistinguishable for the verifier. Furthermore, it guarantees that multiple proofs of possession of even the same signature (from the same signer) remain unlinkable for the verifier.

Flexible Threshold: To achieve a flexible threshold, one can use either or a combination of the following two techniques based on the application at hand: (i) designing multiple schemes with different thresholds and (ii) using *dummy* attributes. The latter is specifically useful to enable thresholds less than the scheme threshold d and it works as follows. The key generation authority assumes that all signers in the system possess some dummy attributes and generates their signing key components accordingly. To enforce a verification policy with threshold $d' \leq d$, a verifier includes $d - d'$ dummy attributes in its verification attribute set. Possessing at least d attributes out of the "new" verification attribute set is equivalent to possessing at least d' attributes out of the "original" verification attribute set, since the $d - d'$ dummy attributes are possessed by all signers. Hence, a range of thresholds $[d_{\min} .. d_{\max}]$ can be supported by a scheme with $d_{\max} - d_{\min}$ dummy attributes at the price of increasing the size of all signing keys and signatures by $d_{\max} - d_{\min}$ components.

3.3 Constructions

We propose a threshold attribute-based signature based on bilinear maps. We make use of some design techniques of earlier works of [SW05] and [BB04].

The Scheme: Signer attributes are assumed to be sets of at most n elements of $\mathbb{Z}_p$. Although generally, identities can be sets of at most n arbitrary strings and a collision resistant hash function is used to map the strings to elements of $\mathbb{Z}_p$. We use $N = \{1, 2, \ldots, n+1\}$ to denote the set of possible attributes. In the following, a basic scheme with a fixed threshold d is introduced. We will later discuss how to extend our scheme for verifiers with different thresholds. Let $\mathbb{G}_1 = \langle g \rangle$ be a group of prime order p and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be defined. Let $(e, \mathbb{G}_1, \mathbb{G}_2)$ be of public knowledge. We present the scheme for signing messages in $\mathbb{Z}_p$. Although, the message space can be expanded to contain arbitrary messages using a collision resistant hash function to map strings to $\mathbb{Z}_p$.

Setup(1^k): Pick y randomly from $\mathbb{Z}_p$ and set $g_1 = g^y$. Pick random elements g_2, h, t_1, t_2, ..., t_{n+1} from

$\mathbb{G}_1$. Define and output the following:

$$T(x) \triangleq g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)} \quad \text{and} \quad msk = y \quad \text{and} \quad mpk = (g, g_1, g_2, t_1, t_2, \ldots, t_{n+1}, h)$$

$\mathsf{KeyGen}(msk, A)$: Choose a random $d-1$ degree polynomial $q(x)$ such that $q(0) = y$, choose random elements r_i in $\mathbb{Z}_p$ for $i \in A$, and output

$$ssk = \Big\langle \; \Big\{ \; g_2^{q(i)} T(i)^{r_i}, \quad g^{r_i} \; \Big\}_{i \in A} \; \Big\rangle$$

$\mathsf{Sign}(ssk, m)$: Parse the signing key as $ssk = \big\langle \{ssk_{1i}, ssk_{2i}\}_{i \in A} \big\rangle$, pick random elements s_i in $\mathbb{Z}_p$ for all $i \in A$, and output

$$\sigma = \big\langle \quad A, \quad \{ \quad ssk_{1i} \, (g_1^m \cdot h)^{s_i}, \quad ssk_{2i}, \quad g^{s_i} \quad \}_{i \in A} \quad \big\rangle$$

$\mathsf{Verify}(mpk, m, \sigma, B)$: Parse the signature as $\sigma = \big\langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \big\rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$ and check if the following equation holds:

$$\prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m \cdot h, \sigma_{3i})} \right)^{\Delta_{i,S}(0)} = e(g_2, g_1) \tag{1}$$

Correctness and Unforgeability: One can easily check the correctness of the scheme by verifying that the S components of the signature *interpolate* to result in $e(g_2, g_1)$. The correctness proof can be found in Appendix B for completion. Unforgeability is implied by converted-signature unforgeability that we prove later in Theorem 1.

Additional Protocols: The concrete conversion and converted-signature verification algorithms are as follows:

$\mathsf{Convert}(mpk, m, \sigma, B)$: Parse the signature as $\sigma = \big\langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \big\rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$. Calculate the converted signature components as follows:

$$\begin{array}{llll} \text{for all } i \in S: & \tilde{\sigma}_{1i} \leftarrow \sigma_{1i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow \sigma_{2i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow \sigma_{3i}^{1/\Delta_{i,B \setminus S}(0)} \\ \text{for all } i \in B \setminus S: & \tilde{\sigma}_{1i} \leftarrow (T(i) g_1^m h)^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} \end{array}$$

and output $\tilde{\sigma} = \big\langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \big\rangle$.

$\mathsf{CvtVerify}(mpk, m, \tilde{\sigma}, B)$: Parse the converted signature as $\tilde{\sigma} = \big\langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \big\rangle$. Check if the following equation holds:

$$\prod_{i \in B} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(g_1^m \cdot h, \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} = e(g_2, g_1) \tag{2}$$

Furthermore, the $\mathsf{iVerify}$ protocol flow is as follows:

1. The signature holder randomizes the converted signature by first choosing random elements s'_i and r'_i for $i \in B$ and then calculating the following. Note that the resulting randomized converted signature is a valid converted signature itself.

$$\breve{\sigma}_{1i} = \tilde{\sigma}_{1i} \cdot T(i)^{r'_i} \, (g_1^m \cdot h)^{s'_i} \quad \text{and} \quad \breve{\sigma}_{2i} = \tilde{\sigma}_{2i} \cdot g^{r'_i} \quad \text{and} \quad \breve{\sigma}_{3i} = \tilde{\sigma}_{3i} \cdot g^{s'_i}$$

2. The signature holder chooses random values τ_i for all $i \in B$ and sets $\hat{\sigma}_{1i} \leftarrow \breve{\sigma}_{1i}^{1/\tau_i}$ and sends $\big\langle \{\hat{\sigma}_{1i}, \breve{\sigma}_{2i}, \breve{\sigma}_{3i}\}_{i \in B} \big\rangle$ to the verifier.

3. Both the signature holder and the verifier calculate the following for all $i \in B$:

$$\begin{array}{ll} u_0 \leftarrow e(g_2, g_1) & u_{1i} \leftarrow e(\hat{\sigma}_{1i}, g)^{\Delta_{i,B}(0)} \\ u_2 \leftarrow \prod_{i \in B} e(T(i), \breve{\sigma}_{2i})^{\Delta_{i,B}(0)} & u_3 \leftarrow \prod_{i \in B} e(g_1^m h, \breve{\sigma}_{3i})^{\Delta_{i,B}(0)} \end{array}$$

4. The signature holder performs the following ZK-PoK for the verifier:

$$\mathsf{ZK\text{-}PoK}\{ \quad (\{\tau_i\}_{i \in B}) : \quad \prod_{i \in B} u_{1i}^{\tau_i} = u_0 u_2 u_3 \quad \}$$

Correctness and Unforgeability: The t-ABS$^+$ scheme is correct since (i) the t-ABS scheme is correct, and (ii) the S components of the converted signature interpolate to result in $e(g_2, g_1)$ and the $B \setminus S$ components are *dummy* components, and (iii) the equation checked in the iVerify protocol is only a rearranged version of what is checked in the CvtVerify algorithm. We discuss unforgeability of our scheme, and in particular, the proof of the following theorem, in Appendix C.

Theorem 1 *The above t-ABS$^+$ scheme is* C-SUF-CMAA*-secure if the CDH problem is hard. As a direct corollary, the underlying t-ABS scheme is* SUF-CMAA*-secure if the CDH problem is hard.*

Full Security: Existential unforgeability can be achieved in either of the following techniques:

General Reduction: Any selectively unforgeable t-ABS can be proved to be existentially unforgeable. This general reduction is discussed in Appendix A. This reduction is not efficient and introduces a large penalty factor to the security of the scheme, but it is carried out in the standard model.

Random Oracles: An alternative method is use random oracles to hash messages and signer attributes at the time of signing. This method is discussed in Appendix A and provides an efficient reduction. The penalty factor here is substantially lower than that of the general reduction.

Waters' Technique: Waters proposed a technique [Wat05] that can be used here to achieve a *tight* existential unforgeability reduction in the standard model at the price of large public keys. To use this technique, we needs to (i) add ℓ random elements $h_1, h_2, \ldots, h_\ell$ from $\mathbb{G}_1$ to mpk in Setup algorithm, where ℓ is the (maximum) bit length of messages, and (ii) replace all instances of $g_1^m h$ in the above scheme with $\mathrm{W}(m)$, where Waters function $\mathrm{W}(\cdot)$ is defined as $\mathrm{W}(m) = h \prod h_i^{m_i}$, where m_i denotes the i-th bit of m. The concrete scheme is transcribed in Appendix F for completeness.

Of course, Waters' technique is preferred to the other two. However, we use the algebraic properties of our basic scheme to construct C-signatures in the next section and we do not know if C-signatures can be constructed based on the Waters modification of our basic t-ABS scheme. Thus, our construction of C-signatures admits only to the first two techniques.

Signer-Attribute Privacy: One can see that since $B \setminus S$ components of the converted signature are publicly simulatable, weak signer-attribute privacy is achieved. Furthermore, the iVerify protocol can be easily proved to be both a proof of knowledge and zero knowledge and hence full signer-attribute privacy is also achieved. We prove the following theorem in Appendix E:

Theorem 2 *The above t-ABS$^+$ scheme achieves both weak and full signer-attribute privacy.*

Efficiency: The iVerify protocol is of size linear in the size of the verification attribute set. This is in contrast with the discussed partial proofs of knowledge, which require proofs of size linear in the number of possible signers.

4 Threshold Attribute-Based C-Signatures

We define threshold attribute-based C-signatures to accommodate for construction of an anonymous credential scheme where users' pseudonyms are in the form of committed values. Hence, we extend t-ABS$^+$

schemes to schemes supporting efficient protocols for signing and verifying signatures on committed values (i.e., pseudonyms) in the following.

4.1 Definition

A t-ABCS scheme consists of a t-ABS$^+$ scheme defined on messages in the form $\tilde{m} = (m, r)$, a commitment scheme, and the following additional protocols:

iCSign: An *interactive signing protocol* for signing a committed value on the *signer* side and obtaining a signature on a committed value on the *user* side. The public inputs are the t-ABS master public key *mpk*, the commitment public key *cpk*, and a commitment M. User's private inputs are the message to be signed (m, r) containing a random value r such that $M = \mathsf{Commit}(cpk, m, r)$. Signer's private input is its signing key *ssk*. At the end of the protocol execution the user will output signer's signature σ on m.

$\mathsf{iHVerify}$: An *interactive verification protocol* for proving knowledge of a converted signature on the *prover* side and verifying a converted signature on the *verifier* side. The public inputs are the t-ABS master public key *mpk* and verifier's verification attribute set B. Prover's private input is the tuple $(m, r, \tilde{\sigma})$ such that $\tilde{\sigma}$ is a converted signature on (m, r). The verifier has no private input. At the end of the protocol execution the verifier will output a binary value reflecting prover's converted signature validity against B.

$\mathsf{iCVerify}$: An *interactive verification protocol* for proving knowledge of a converted signature on a committed value on the *prover* side and verifying a converted signature on a committed value on the *verifier* side. The public inputs are the t-ABS master public key *mpk*, the commitment public key *cpk*, a commitment M', and verifier's verification attribute set B. Prover's private input is the tuple $(m, r, r', \tilde{\sigma})$ such that $M' = \mathsf{Commit}(cpk, m, r')$ and $\tilde{\sigma}$ is a converted signature on (m, r). The verifier has no private input. At the end of the protocol execution the verifier will output a binary value reflecting prover's converted signature validity against B.

Correctness: Besides correctness of the underlying t-ABS$^+$ scheme, we require that the above protocols should satisfy the correctness property. For iCSign it means that if the user and the signer follow the protocol, then the user should output a valid signature. In other words, if σ is the output the user gets from running the protocol with a signer with attribute set A, then $\mathsf{Convert}(mpk, (m, r), \sigma, B)$ should pass the verification test for any B if $|A \cap B| \geq d$. Correctness for $\mathsf{iHVerify}$ and $\mathsf{iCVerify}$ means that if the prover and the verifier follow the protocol, then the verifier finds out the validity or invalidity of the prover's converted signature correctly. This means that the verifier's outputs is the same as $\mathsf{CvtVerify}(mpk, (m, r), \tilde{\sigma}, B)$.

Security: Besides security of the underlying t-ABS$^+$ scheme, we require the following security properties from the additional protocols. We require that the iCSign protocol is secure in the following senses:

Security for the user: Users with different private inputs m should remain indistinguishable for the signer, even if the signer acts maliciously. In particular, we require that for any *mpk* and *cpk*, there is no malicious signer that can distinguish if it is interacting with a user with private input containing m_0 or with a user with private input containing m_1.

Security for the signer: The protocol should reveal (almost) no information other than a single signature on a known committed value to a user, even though the user acts maliciously. In particular, we require that for any *mpk* and *cpk*, and for any (possibly malicious) user, there exists a *simulator*, that with only a *one-time* access to the signing oracle, can simulate a signer's interaction with the user.

We require that the $\mathsf{iHVerify}$ protocol is (i) a zero knowledge protocol (*security for the prover*) and (ii) a proof of knowledge of a triplet $(m, r, \tilde{\sigma})$ such that $\mathsf{CvtVerify}(mpk, (m, r), \tilde{\sigma}, B) = 1$ (*security for the verifier*). We require that the $\mathsf{iCVerify}$ protocol is (i) a zero knowledge protocol (*security for the* *prover*) and (ii) a proof of knowledge of a quadruple $(m, r, r', \tilde{\sigma})$ such that $M' = \mathsf{Commit}(cpk, m, r')$ and $\mathsf{CvtVerify}(mpk, (m, r), \tilde{\sigma}, B) = 1$ (*security for the verifier*).

A t-ABCS scheme is said to be correct and secure if the underlying t-ABS$^+$ scheme defined on messages in the form $\tilde{m} = (m, r)$, is correct and C-EUF-CMAA-secure, the commitment scheme is correct and secure, and the associated iCSign, iHVerify and iCVerify protocols are correct and secure for the user, signer, prover and verifier in the above senses.

4.2 Construction

Our t-ABS$^+$ scheme can be modified to be defined on messages in the form of $\tilde{m} = (m, r)$ as follows: (i) in the Setup algorithm, add a random element g_3 from $\mathbb{G}_1$ to the master public key and set $mpk = (g, g_1, g_2, g_3, t_1, t_2, \ldots, t_{n+1}, h)$, and (ii) in the Sign, Verify, Convert, and CvtVerify algorithms, replace g_1^m with $g_1^m g_3^r$ everywhere. We will not use the iVerify protocol. The scheme is provided in Appendix D for completion.

For the commitment scheme, consider the scheme with $cpk = (g_1, g_3)$ and $\mathsf{Commit}(cpk, m, r) = g_1^m g_3^r$. This scheme is (unconditionally) hiding and (computationally) binding if the discrete logarithm problem is hard. For this commitment scheme and the above t-ABS scheme, we introduce the following additional protocols to construct a t-ABCS scheme altogether. The iCSign protocol is as follows:

1. the user gives a ZK-PoK of (m, r) such that $M = g_1^m \cdot g_3^r$.
2. the signer picks random elements s_i in $\mathbb{Z}_p$ for all $i \in A$, calculates the signature as below, and sends it to the user.
$$\sigma = \Big\langle \quad A, \quad \{ \quad ssk_{1i}(Mh)^{s_i}, \quad ssk_{2i}, \quad g^{s_i} \quad \}_{i \in A} \quad \Big\rangle$$

The iHVerify and iCVerify protocols are as follows:

1. The signature holder randomizes the converted signature by first choosing random elements s'_i and r'_i for $i \in B$ and then calculating the following. Note that the resulting randomized converted signature is a valid converted signature itself.
$$\breve{\sigma}_{1i} = \tilde{\sigma}_{1i} \cdot T(i)^{r'_i} \, (g_1^m g_3^r \cdot h)^{s'_i} \quad \text{and} \quad \breve{\sigma}_{2i} = \tilde{\sigma}_{2i} \cdot g^{r'_i} \quad \text{and} \quad \breve{\sigma}_{3i} = \tilde{\sigma}_{3i} \cdot g^{s'_i}$$
2. The signature holder chooses random values τ_i for all $i \in B$ and sets $\hat{\sigma}_{1i} \leftarrow \breve{\sigma}_{1i}^{1/\tau_i}$ and sends $\big\langle \{\hat{\sigma}_{1i}, \breve{\sigma}_{2i}, \breve{\sigma}_{3i}\}_{i \in B} \big\rangle$ to the verifier.
3. Both the signature holder and the verifier calculate the following for all $i \in B$:
$$\begin{array}{lll} u_0 \leftarrow e(g_2, g_1) & u_{1i} \leftarrow e(\hat{\sigma}_{1i}, g)^{\Delta_{i,B}(0)} & u_2 \leftarrow \prod_{i \in B} e(T(i), \breve{\sigma}_{2i})^{\Delta_{i,B}(0)} \\ u_{31} \leftarrow \prod_{i \in B} e(g_1, \breve{\sigma}_{3i})^{\Delta_{i,B}(0)} & u_{32} \leftarrow \prod_{i \in B} e(g_3, \breve{\sigma}_{3i})^{\Delta_{i,B}(0)} & u_{33} \leftarrow \prod_{i \in B} e(h, \breve{\sigma}_{3i})^{\Delta_{i,B}(0)} \end{array}$$
4. The signature holder performs the following ZK-PoK for the verifier based on the protocol:
$$\begin{array}{ll} \text{for } \mathsf{iHVerify}: & \mathsf{ZK\text{-}PoK}\big\{ \quad (m, r, \{\tau_i\}_{i \in B}) : \quad \prod_{i \in B} u_{1i}^{\tau_i} = u_0 u_2 u_{31}^m u_{32}^r u_{33} \quad \big\} \\ \text{for } \mathsf{iCVerify}: & \mathsf{ZK\text{-}PoK}\big\{ \quad (m, r, r', \{\tau_i\}_{i \in B}) : \quad M' = g_1^m g_3^{r'} \quad \wedge \quad \prod_{i \in B} u_{1i}^{\tau_i} = u_0 u_2 u_{31}^m u_{32}^r u_{33} \quad \big\} \end{array}$$

Correctness: In Appendix E we briefly show that the above protocols are correct.

Security: On unforgeability of our scheme, in Appendix C we prove Theorem 3 that comes in the following. As discussed under unforgeability of our t-ABS scheme, the theorem implies that the scheme is also EUF-CMAA-secure with a loose reduction or is EUF-CMAA-secure with an efficient reduction in the random oracle model. On security of our additional protocols, we prove the Theorem 4 that comes in the following in Appendix E. Furthermore, hardness of the CDH problem implies hardness of the discrete logarithm problem, which, in turn, is sufficient for the commitment scheme to be secure. Thus our t-ABCS scheme is secure as per our definition if CDH is hard.

Theorem 3 *The above t-ABCS scheme, i.e. our t-ABS$^+$ scheme defined on messages in the form $\tilde{m} = (m, r)$, is* C-SUF-CMAA*-secure if the CDH problem is hard.*

Theorem 4 *The above protocols* iCSign*,* iHVerify*, and* iCVerify *are secure for the user, signer, prover and verifier.*

5 Threshold Attribute-Based Anonymous Credential Systems

A credential system involves users and organizations. Organizations issue *credentials* to users based on their policy, which might require users to prove possession of credentials from other organizations. Users are considered holders of the credential they are issued with and might wish to prove possession of their credentials to organizations to get a service or be granted new credentials. An *anonymous credential system* is one in which users are known to organizations only by their *pseudonyms*. Such pseudonyms should be well-defined, i.e., each pseudonyms must belong to only one user.

A basic anonymous credential system must support a minimal of three basic protocols, respectively for *forming* pseudonyms, *granting* credentials, and *verifying* credentials. All these protocols are between a user and an organization. Users first form the pseudonyms with any organization that they might wish to issue a credential to them. Then they might be granted a credential on their formed pseudonym. Possession of such a credential can be verified by an organization at a later time. The user proves possession of her credential to obtain a service from the verifier. If this service includes issuing a credential to the user, then the user must have formed a pseudonym with the organization previously and prove possession of her credential on the formed pseudonym. Otherwise, there is no need to involve a pseudonym. The user simply proves possession of her credential in this case. Thus, an optional property is that the system supports two types of verification protocols: one for verifying possession of a credential on a formed pseudonym, and another for simply verifying possession of a credential, which implements more efficiently than the former.

An attribute-based anonymous credential system is one with attribute-based organizations. Each organization is given a signing key based on its attributes by a trusted authority. A threshold attribute-based anonymous credential system is an attribute-based anonymous credential system that supports threshold attribute-based verification. We assume that the "credential types" in the system are fixed and each organization is given extra attributes for the credential types they can issue. Hence, a credential only consists of the identity of the credential holder and the signature of the organization on it, using the appropriate signing attribute for the credential type. Hence, a driving license would be in the form of a signature on the identity of a user with the attribute "authorized driving license issuer" among the issuer organization attribute set.

5.1 Security Framework

To formalize a security definition for a threshold attribute-based anonymous credential system, we use the simulation paradigm. Two models of a credential system, an ideal model and a real model, are presented. In both models, the system contains users and organizations as entities. Each model contains an *active* adversary in the sense that not only the adversary sees all the messages sent to the *corrupted* entities (users and organizations), but also upon corruption, the adversary gets hold of all the secret information and future actions of the corrupted entity. However, only *static* adversaries are considered. That is, the adversary only gets to pick the entities it wishes to corrupt in the beginning of the system execution and is not allowed to dynamically corrupt new entities during the system run. In the real model, the entities communicate with each other directly. However, in the ideal model, they communicate via a *trusted party*, which locally performs all calculations needed to carry out a transaction and just reports the corresponding outputs to the entities involved. The adversary is assumed to be unable to eavesdrop the communications. To be able to compare the two systems, a scheduler entity is introduced to the system, called the *environment*. The environment schedules transactions in the system and gathers output information. In particular, it tells each entity which protocol to carry out and with whom. At the end of the protocol, the entities involved report back the outcome of the protocol. The environment proceeds in periods and in each period it schedules only one transaction. Concurrent scheduling is not considered[2]. A depiction of a typical system with the above setting is shown in Figure 1. This framework of security is the same as what Camenisch and Lysyanskaya used for defining security of anonymous credential systems [CL01, Lys02].

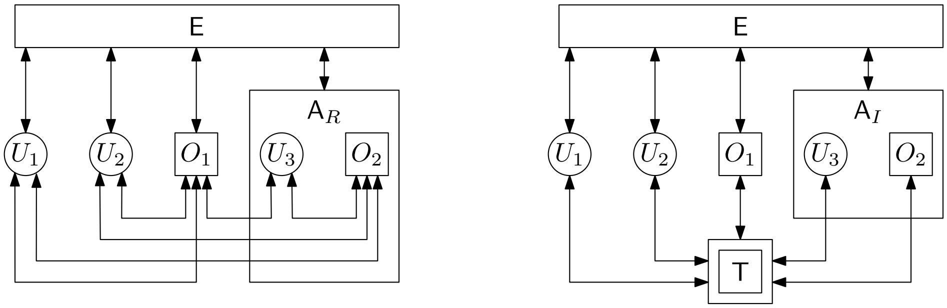

Figure 1: The real model (left) vs. the ideal model (right) in a system consisting of the environment E, the users U_1, U_2, and U_3, and the organizations O_1 and O_2, where the adversary A controls the user U_3 and the organization O_2. T is the ideal-world trusted party.

A real system is said to be secure if it *emulates* a secure ideal model. Emulation means that for any arbitrary scheduling and any arbitrary real-model adversary, an ideal-model adversary can be found such that the two systems are indistinguishable in the eyes of the environment. An implication of this property is that whatever the real-model adversary can extract from the real-model system, the ideal-model adversary can extract from the ideal-model system. Now, since the ideal system is defined in a way that its security against ideal-model adversaries is guaranteed, the security of the real-model system follows.

5.2 Ideal Model for t-ABACS

The Model: In the ideal, model users and organizations interact through a trusted party T who makes sure the system remains anonymous and secure. There is a public universal set of attributes $\mathbb{U}$. Each organization O has a set of attributes $A_O \in \mathbb{U}$, which is assumed to be public and fixed during the life of the system. Besides, each organization O, based on its policy at a certain time, might be interested to verify users' credentials against a set of attributes. We denote this set by $B_O \in \mathbb{U}$. B_O is assumed to be made public by the organization and fixed during each period. T maintains three lists: a list of user passwords L_{P}, a list of user pseudonyms L_{N}, and a list of issued credentials L_{C}. L_{P} contains pairs of user names U and their passwords K_U and is used to authenticate (and hence identify) users. We assume that initially this list contains all user names and passwords in the system. L_{N} contains triplets of user names U, organization names O, and their pseudonyms N_{UO}, and is initially empty and is filled by T as users form pseudonyms with organizations. L_{C} contains pairs of user pseudonyms N_{UI} and issuer organization attributes A_I, and is initially empty and is filled by T as organizations grant credentials to users. User pseudonyms are assumed to be chosen randomly and independent of user identities. There are four basic operations as follows.

[2] Note that this notion of security is weaker than Canetti's *Universal Composability* [Can01]. We do not address a *composable* notion of security, but rather a *stand-alone* one.

$\mathsf{FormNym}\,[U \leftrightarrow \mathsf{T} \leftrightarrow O]\,(N_{UO})$:
This is a protocol between a user U and an organization O. U wants to form a pseudonym N_{UO} with O. The protocol flow is as follows.

$U \mathrel{-\!\!(} \texttt{FormNym}, (U, K_U), N_{UO}, O \mathrel{)\!\!\rightarrow} \mathsf{T}$:
U sends her login info (U, K_U) to T along with a request for forming a pseudonym N_{UO} with O.

$\mathsf{T} \mathrel{-\!\!(} \texttt{FormNym}, N_{UO} \mathrel{)\!\!\rightarrow} O \quad \text{if} \quad (U, K_U) \in L_{\mathrm{P}}$:
T checks U's login info and if K_U is U's password then tells O that a user wants to form a pseudonym N_{UO} with it.

$O \mathrel{-\!\!(} d \mathrel{)\!\!\rightarrow} \mathsf{T}$:
O either accepts or rejects the request and informs T of its decision d.

$\mathsf{T} \mathrel{-\!\!(} d \mathrel{)\!\!\rightarrow} U$:
T informs U of O's decision d. If O accepts, T also stores the triple (U, O, N_{UO}) in L_{N}.

$\mathsf{GrantCred}\,[U \leftrightarrow \mathsf{T} \leftrightarrow I]\,(N_{UI})$:
This is a protocol between a user U and an organization I. U, who has already formed a pseudonym N_{UI} with I, wants to be granted a credential by I on N_{UI}. I has set of attributes A_I. The protocol flow is as follows.

$U \mathrel{-\!\!(} \texttt{GrantCred}, (U, K_U), N_{UI}, I \mathrel{)\!\!\rightarrow} \mathsf{T}$:
U sends her login info (U, K_U) to T along with a request for being granted a credential by I on a pseudonym N_{UI}.

$\mathsf{T} \mathrel{-\!\!(} \texttt{CrantCred}, N_{UI} \mathrel{)\!\!\rightarrow} I \quad \text{if} \quad (U, K_U) \in L_{\mathrm{P}} \ \wedge \ (U, I, N_{UI}) \in L_{\mathrm{N}}$:
T checks U's login and pseudonym info. If K_U is U's password and N_{UI} is U's pseudonym with I then T tells I that the user with pseudonym N_{UI} wishes to be granted a credential.

$I \mathrel{-\!\!(} d \mathrel{)\!\!\rightarrow} \mathsf{T}$:
I either accepts or rejects the request and informs T of its decision d.

$\mathsf{T} \mathrel{-\!\!(} d \mathrel{)\!\!\rightarrow} U$:
T informs U of I's decision d. If I accepts, T also stores the pair (N_{UI}, A_I) in L_{C}.

$\mathsf{VerifyCred}\,[U \leftrightarrow \mathsf{T} \leftrightarrow V]\,(N_{UI})$:
This is a protocol between a user U and an organization V. U has a credential issued by I on N_{UI}. A_I is the attribute set of I and B_V is the verification attribute set of V. U wants V to verify that she has been issued a credential by an organization whose attributes A_I has at least d elements in common with V's verification attribute set B_V. The protocol flow is as follows.

$U \mathrel{-\!\!(} \texttt{VerifyCred}, (U, K_U), N_{UI}, V \mathrel{)\!\!\rightarrow} \mathsf{T}$:
U sends her login info (U, K_U) to T along with a request for her credential on N_{UI} to be verified by V.

$\mathsf{T} \mathrel{-\!\!(} \texttt{VerifyCred} \mathrel{)\!\!\rightarrow} V \quad \text{if} \quad (U, K_U) \in L_{\mathrm{P}} \ \wedge \ \big[\, \exists I : (U, I, N_{UI}) \in L_{\mathrm{N}} \ \wedge \ (N_{UI}, A_I) \in L_{\mathrm{C}} \ \wedge \ |A_I \cap B_V| \geq d \,\big]$:
T checks U's login and credential info. If K_U is U's password and there exists an organization I such that N_{UI} is U's pseudonym with I and a credential has been issued on N_{UI} by an organization with attributes A_I which has at least d elements in common with V's verification attribute set B_V, then T tells I that the user has a credential from an organization whose attributes has at least d elements in common with V's verification attribute set.

$V \mathrel{-\!\!(} \texttt{Ack} \mathrel{)\!\!\rightarrow} \mathsf{T}$:
V acknowledges verification of the credential.

$\mathsf{T} \mathrel{-\!\!(} \texttt{Ack} \mathrel{)\!\!\rightarrow} U$:
T passes V's acknowledgment to U.

$\mathsf{VerifyCredOnNym}\,[U \leftrightarrow \mathsf{T} \leftrightarrow V]\,(N_{UI}, N_{UV}):$ This is a protocol between a user U and an organization V. U has a credential issued by I on N_{UI} and has already formed a pseudonym N_{UV} with V. A_I is the attribute set of I and B_V is the verification attribute set of V. U wants V to verify that she has been issued a credential by an organization whose attributes A_I has at least d elements in common with V's verification attribute set B_V. The protocol flow is as follows.

$U \dashv\!\!(\, \texttt{VerifyCredOnNym}, (U, K_U), N_{UI}, V, N_{UV} \,)\!\!\rightarrowtail \mathsf{T}$:
U sends her login info (U, K_U) to T along with a request for her credential on N_{UI} to be verified by V who knows her by N_{UV}.

$\mathsf{T} \dashv\!\!(\, \texttt{VerifyCredOnNym}, N_{UV} \,)\!\!\rightarrowtail V$ if $(U, K_U) \in L_{\mathrm{P}} \;\wedge\; (U, V, N_{UV}) \in L_{\mathrm{N}} \;\wedge\; \big[\, \exists I : (U, I, N_{UI}) \in L_{\mathrm{N}} \;\wedge\; (N_{UI}, A_I) \in L_{\mathrm{C}} \;\wedge\; |A_I \cap B_V| \geq d \,\big]$:
T checks U's login, nym, and credential info. If K_U is U's password, N_{UV} is U's pseudonym with V, and there exists an organization I such that N_{UI} is U's pseudonym with I and a credential has been issued on N_{UI} by an organization with attributes A_I which has at least d elements in common with V's verification attribute set B_V, then T tells I that the user with pseudonym N_{UV} has a credential from an organization whose attributes has at least d elements in common with V's verification attribute set.

$V \dashv\!\!(\, \texttt{Ack} \,)\!\!\rightarrowtail \mathsf{T}$:
V acknowledges verification of the credential.

$\mathsf{T} \dashv\!\!(\, \texttt{Ack} \,)\!\!\rightarrowtail U$:
T passes V's acknowledgment to U.

Usage Scenario: Let's show how the above ideal model can be used to provide an attribute-based and anonymous solution for a credential system. Consider a user U in the system who already has credentials from organizations O_1 and O_2 on her pseudonyms with them, N_{UO_1} and N_{UO_2}, respectively. There is a third organization O_3 that issues credentials to users that possess two credentials: (i) a credential from an organization with at least d attributes in B_{31}, and (ii) a credential from an organization with at least d attributes in B_{32}. Suppose that O_1 and O_2 satisfy these conditions, respectively. Now, to get a credential from O_3, U first runs $\mathsf{FormNym}(N_{UO_3})$ with O_3 to form the pseudonym N_{UO_3}. Then, she runs $\mathsf{VerifyCredOnNym}(N_{UO_1}, N_{UO_3})$ with verification attribute set B_{31} and $\mathsf{VerifyCredOnNym}(N_{UO_2}, N_{UO_3})$ with verification attribute set B_{32} with O_3. As a result, O_3 is convinced that the owner of pseudonym N_{UO_3} satisfies conditions (i) and (ii), and hence issues a credential on the same pseudonym by running $\mathsf{GrantCred}(N_{UO_3})$ with her. Note that, O_3 does not learn anything more than conditions (i) and (ii) about O_1 and O_2. Suppose there is a fourth organization O_4 that provides services (that does not include issuing credentials) to users that possess a credential from an organization with at least d attributes in B_4. U can simply run $\mathsf{VerifyCred}(N_{UO_3})$ with verification attribute set B_4 with O_4. This convinces O_4 that the user they are dealing with satisfies their policy. Hence, the service is provided to the user.

Properties of the Model: The above model ensures the following security and privacy properties:

Attribute-Based Unforgeability of Credentials: The fact that records can be added to L_{C} only when a credential is issued together with checking L_{C} at verification time make it impossible to forge credentials. That is, one cannot prove a credential via $\mathsf{VerifyCred}$ or $\mathsf{VerifyCredOnNym}$ if it was not issued by an issuer with at least d attributes in common with B.

Privacy of Users: Organizations do not find out anything about a user other than her ownership of some credentials, even if they collude. Thus, the users' privacy is preserved except for the inherent and unavoidable leakage of information as a result of obtaining and/or verification of credentials under the same pseudonym. This property includes the following properties:

Anonymity: The only pieces of information about users the organizations see during the protocols are their pseudonyms. Hence, users are known to organizations only by their pseudonyms and to link a pseudonym to a user, even if organizations and other users collude, they cannot do better than random guessing.

Unlinkability: It is infeasible to link two transactions involving the same user as long as the user uses different pseudonyms in the transactions. Hence, it is also infeasible to link two pseudonyms belonging to the same user.

Non-transferability: Since L_{N} is checked during both issuing and verifying the credentials, it is infeasible to get credentials on behalf of some other user or transfer one's credential to others.

Issuer Attribute Privacy: Verifiers do not find out anything about the attributes of the credential issuers other than the fact that their attributes satisfy the verification policy.

5.3 A Concrete t-ABACS System

Consider a t-ABCS scheme as defined above. We propose the following t-ABACS system based on this scheme. We assume that there is a trusted signing key generator authority outside the system that issues signing keys for organizations based on their attributes. Organizations' attribute sets are all subsets of a universal public attribute set $\mathbb{U}$.

$\mathsf{Init}(1^k)$: During the initiation phase, each user U in the system picks a secret SK_U and each organization O with attributes A_O contacts the system signing key generator to get a signing secret key ssk_O.

$\mathsf{FormNym}\,[U \leftrightarrow O]\,(N_{UO})$: User U picks a random r_O and forms a commitment to her secret $N_{UO} = \mathsf{Commit}(cpk, SK_U, r_O)$. She then sends N_{UO} to O and proves that she knows the pair (SK_U, r_O) using a ZK-PoK protocol. U and O save N_{UO} as the pseudonym of U with O.

$\mathsf{GrantCred}\,[U \leftrightarrow I]\,(N_{UI})$: User U and credential issuer I who have already formed a pseudonym $N_{UI} = \mathsf{Commit}(cpk, SK_U, r_I)$, carry out the iCSign protocol on public inputs (mpk, cpk, N_{UI}), user's private input (SK_U, r_I), and issuer's private input ssk_I. User stores her output σ.

$\mathsf{VerifyCred}\,[U \leftrightarrow V]\,(N_{UI})$: User U first calculates $\tilde{\sigma} = \mathsf{Convert}(mpk, (SK_U, r_I), \sigma, B_V)$. Then, user U and verifier V carry out the $\mathsf{iHVerify}$ protocol on public inputs (mpk, B_V) and user's private inputs $(SK_U, r_I, \tilde{\sigma})$.

$\mathsf{VerifyCredOnNym}\,[U \leftrightarrow V]\,(N_{UI}, N_{UV})$: User U first calculates $\tilde{\sigma} = \mathsf{Convert}(mpk, (SK_U, r_I), \sigma, B_V)$. Then, user U and verifier V who have already formed a pseudonym $N_{UV} = \mathsf{Commit}(cpk, SK_U, r_V)$, carry out the $\mathsf{iCVerify}$ protocol on public inputs (mpk, cpk, N_{UV}, B_V) and user's private inputs $(SK_U, r_I, r_V, \tilde{\sigma})$.

We assume that pseudonyms are well-defined. That is, each pseudonym belongs to only one user. In practice, this can be guaranteed as follows. All users are required to reveal their identity and pseudonyms to a trusted authority and get a *zeroth credential* for each of their pseudonyms. Possession of this zeroth credential is proved to an organization at the time of forming the pseudonym with the organization.

We prove that the above system is a secure implementation of the t-ABACS system. In particular, in Appendix G we prove the following theorem. In our proof, we provide an ideal-model adversary for any real-model adversary and briefly show how they remain indistinguishable in the eyes of the environment as long as the underlying t-ABCS scheme is secure.

Theorem 5 *The above concrete t-ABACS system emulates the ideal model for a t-ABACS if the underlying t-ABCS scheme is* EUF-CMAA-*secure.*

6 Conclusions and Open Problems

We introduced a new scheme called a threshold attribute-based signature, which allows verification of signatures as originating from a fuzzy signer. We proposed a basic concrete t-ABS scheme and provided a tight security reduction for selective unforgeability based on hardness of the CDH problem and discussed how to achieve existential unforgeability in both the standard and the random oracle models. We have shown that our basic t-ABS scheme admits to an efficient threshold attribute-based C-signature that can be used as a building block to realize privacy-enhanced anonymous credential systems. However, existential unforgeability of our t-ABCS scheme is based on a loose generic reduction. Designing t-ABCS schemes with tight existential unforgeability remains as an open problem.

Our attribute-based signature scheme and hence our attribute-based anonymous credential system only support simple threshold verification policies. A possible future direction is to design schemes that support more complex verification policies. That is, to generalize our schemes to ones in which the verification algorithm gets as input a (complex) policy, rather than a verification attribute set, and the verification goes through if the signer attributes satisfy the verification policy. Systems based on such schemes can accommodate for a larger application spectrum.

References

[BB04] Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004. (Cited on pages 9 and 21.)

[BG92] Mihir Bellare and Oded Goldreich. On Defining Proofs of Knowledge. In Ernest F. Brickell, editor, *CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992. (Cited on page 6.)

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security (ACM-CCS '93)*, pages 62–73. ACM, 1993. (Cited on page 21.)

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS '01*, pages 136–145, 2001. (Cited on page 15.)

[CDM00] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie. Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography (PKC '00)*, volume 1751 of *Lecture Notes in Computer Science*, pages 354–372. Springer, 2000. (Cited on page 6.)

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Yvo Desmedt, editor, *CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994. (Cited on page 4.)

[CE87] David Chaum and Jan-Hendrik Evertse. A Secure and Privacy-protecting Protocol for Transmitting Personal Information Between Organizations. In Andrew M. Odlyzko, editor, *CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 118–167. Springer, 1987. (Cited on page 6.)

[Cha85] David Chaum. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Commun. ACM*, 28(10):1030–1044, 1985. (Cited on page 6.)

[CL01] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In Birgit Pfitzmann, editor, *EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001. (Cited on pages 7 and 15.)

[Cra05] Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus,* *Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005. (Cited on page 20.)

[CS97] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In Burton S. Kaliski Jr., editor, *CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997. (Cited on page 6.)

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, 18(1):186–208, 1989. (Cited on page 6.)

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing But Their Validity for All Languages in NP Have Zero-Knowledge Proof Systems. *J. ACM*, 38(3):691–729, 1991. (Cited on page 7.)

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002. (Cited on page 5.)

[GZ08] Shanqing Guo and Yingpei Zeng. Attribute-based Signature Scheme. *Information Security and Assurance, 2008. ISA 2008. International Conference on*, pages 509–511, April 2008. (Cited on page 5.)

[Kha07a] Dalia Khader. Attribute Based Group Signature with Revocation. Cryptology ePrint Archive, Report 2007/241, 2007. http://eprint.iacr.org/2007/241. (Cited on page 5.)

[Kha07b] Dalia Khader. Attribute Based Group Signatures. Cryptology ePrint Archive, Report 2007/159, 2007. http://eprint.iacr.org/2007/159. (Cited on page 5.)

[Kha08] Dalia Khader. Authenticating with Attributes. Cryptology ePrint Archive, Report 2008/031, 2008. http://eprint.iacr.org/2008/031. (Cited on page 5.)

[LK08] Jin Li and Kwangjo Kim. Attribute-Based Ring Signatures. Cryptology ePrint Archive, Report 2008/394, 2008. http://eprint.iacr.org/2008/394. (Cited on page 5.)

[LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym Systems. In Howard M. Heys and Carlisle M. Adams, editors, *Selected Areas in Cryptography (SAC '99)*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 1999. (Cited on page 6.)

[Lys02] Anna Lysyanskaya. *Signature Schemes and Applications to Cryptographic Protocol Design.* PhD thesis, Massachusetts Institute of Technology, 2002. (Cited on pages 6, 7 and 15.)

[MPR08] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-Based Signatures: Achieving Attribute-Privacy and Collusion-Resistance. Cryptology ePrint Archive, Report 2008/328, 2008. http://eprint.iacr.org/2008/328. (Cited on page 5.)

[Sha84] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO '84*, pages 47–53, 1984. (Cited on page 5.)

[SW05] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In Cramer [Cra05], pages 457–473. (Cited on pages 5 and 9.)

[Wat05] Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Cramer [Cra05], pages 114–127. (Cited on page 11.)

[YCD08] Piyi Yang, Zhenfu Cao, and Xiaolei Dong. Fuzzy Identity Based Signature. Cryptology ePrint Archive, Report 2008/002, 2008. http://eprint.iacr.org/2008/002. (Cited on page 5.)

A Selective and Converted Signature Unforgeability

A weaker notion of security that we consider is *selective unforgeability* against chosen message and attribute set attacks. In particular, we define the following game between a challenger and the adversary.

Initiation Phase: The adversary declares a pair $(\tilde{\mu}, \beta)$ containing the message and the verification attribute set on which it wants to make a forgery.

Setup Phase: The challenger runs the Setup algorithm and gives mpk to the adversary.

Query Phase: The adversary is allowed to ask queries for the following:

- a secret key of a signer with attributes of its choice α as long as $|\alpha \cap \beta| < d$, and
- a signature of a signer with any attribute set of its choice α on a message of its choice $\tilde{m}$ as long as $\tilde{m} \neq \tilde{\mu}$ or $|\alpha \cap \beta| < d$.

Forgery Phase: The adversary outputs a forged converted signature $\tilde{\sigma}$ on the message $\tilde{\mu}$ and the verification attribute set β and wins if $\mathsf{Verify}(mpk, \tilde{\mu}, \tilde{\sigma}, \beta) = 1$.

If no polynomial adversary has a considerable advantage in the above game, we say that the t-ABS scheme is selectively unforgeable against chosen message and attribute set attacks, or SUF-CMAA-secure for short.

It is easy to see that any SUF-CMAA-secure t-ABS scheme is a EUF-CMAA-secure scheme, but the reduction is not efficient. The corresponding security penalty is linear in the product of the sizes of the message and verification attribute set spaces. To see this, consider a successful EUF-CMAA adversary for the scheme. One can use this adversary to SUF-CMAA-break the scheme as follows. First guess the adversary's choice of $(\tilde{\mu}, \beta)$ and declare it as the selected forging target. Then run the adversary. If the guess is correct, the SUF-CMAA attack succeeds and the probability of this event is one over the number of possible messages times the number of possible verification attribute sets. Using the Random Oracle Model [BR93], the security penalty can be reduced to the maximum number of oracle queries the adversary can make. In this case, the random oracle only needs to be 'programmed' in one point, making sure that the forgery target $(\tilde{\mu}, \beta)$ provided by the EUF-CMAA adversary is mapped to the guessed pair. A similar situation in the design of identity-based encryption schemes is elaborated by Boneh and Boyen in [BB04].

The C-EUF-CMAA-security game is defined as follows. The C-SUF-CMAA-security game can be defined accordingly.

Setup Phase: The challenger runs the Setup algorithm and gives mpk to the adversary.

Query Phase: The adversary is allowed to ask queries for the following:

- a secret key of a signer with attributes of its choice α, and
- a signature of a signer with any attribute set of its choice α on a message of its choice m.

Forgery Phase: The adversary outputs a triplet $(\mu, \tilde{\sigma}, \beta)$, consisting of a message μ, a forged converted signature $\tilde{\sigma}$, and a verification attribute set β, and wins if $\tilde{\sigma}$ is a valid converted signature with respect to (mpk, μ, β) and

- for all queried sets of attributes α, we have $|\alpha \cap \beta| < d$, and
- for all queried pairs (α, m), we have $m \neq \mu$ or $|\alpha \cap \beta| < d$.

B Correctness of Our t-ABS$^+$ Scheme

We show that our t-ABS$^+$ scheme satisfies the correctness property. If σ is a correctly produced signature, we have:

$$\begin{aligned}
\prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m \cdot h, \sigma_{3i})} \right)^{\Delta_{i,S}(0)} &= \prod_{i \in S} \left(\frac{e(g_2^{q(i)} T(i)^{r_i} \left(g_1^m \cdot h\right)^{s_i}, g)}{e(T(i), g^{r_i}) \cdot e(g_1^m \cdot h, g^{s_i})} \right)^{\Delta_{i,S}(0)} \\
&= \prod_{i \in S} \left(e(g_2, g)^{q(i)} \right)^{\Delta_{i,S}(0)} = e(g_2, g)^{\sum_{i \in S} q(i) \Delta_{i,S}(0)} \\
&= e(g_2, g)^y = e(g_2, g_1)
\end{aligned}$$

Thus, Equation 1 holds.

Now, note that, from the definition of the Lagrange coefficient, for any subset S of B and all $i \in B$ we have:

$$\Delta_{i,B}(x) = \Delta_{i,S}(x) \cdot \Delta_{i,B \setminus S}(x)$$

Considering this property, if $\tilde{\sigma}$ is a correctly produced converted signature, for the components in S we have:

$$\begin{aligned}
\prod_{i \in S} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(g_1^m \cdot h, \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} &= \prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m \cdot h, \sigma_{3i})} \right)^{\Delta_{i,B}(0) / \Delta_{i,B \setminus S}(0)} \\
&= \prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m \cdot h, \sigma_{3i})} \right)^{\Delta_{i,S}(0)} \\
&= e(g_2, g_1)
\end{aligned}$$

Furthermore, for the components of the converted signature not in S we have:

$$\begin{aligned}
\prod_{i \in B \setminus S} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(g_1^m g_3^r \cdot h, \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} &= \prod_{i \in S} \left(\frac{e(T(i) g_1^m g_3^r h, g)}{e(T(i), g) \cdot e(g_1^m g_3^r \cdot h, g)} \right)^{\Delta_{i,B}(0) / \Delta_{i,B \setminus S}(0)} \\
&= \prod_{i \in S} \left(\frac{e(T(i) g_1^m g_3^r h, g)}{e(T(i), g) \cdot e(g_1^m g_3^r \cdot h, g)} \right)^{\Delta_{i,S}(0)} \\
&= \prod_{i \in S} 1^{\Delta_{i,S}(0)} = 1
\end{aligned}$$

Combining the two results above shows that the verification equation, i.e., Equation 2, holds.

Furthermore, if $\tilde{\sigma}$ is a correctly produced converted signature, one can easily check that the equation

$$\prod_{i \in B} u_{1i}^{\tau_i} = u_0 u_2 u_{31}^m u_{33}$$

in the iVerify protocol, is just a rearrangement of Equation 2 and hence it holds.

C Unforgeability of Our t-ABS$^+$ and t-ABCS Schemes

Theorem 1. Our t-ABS$^+$ scheme is c-suf-cmaa-secure if the CDH problem is hard. As a direct corollary, the underlying t-ABS scheme is suf-cmaa-secure if the CDH problem is hard.

Proof. The proof can be derived from the proof of the Theorem 3 that comes next by setting $r = \rho = 0$. Note that in that case, there would be no type 2 forgery in the following proof, thus the part dealing with a type 2 forger can be ignored. We omit the proof to avoid repetition. ■

Theorem 3. Our t-ABCS scheme, i.e. our t-ABS$^+$ scheme defined on messages in the form $\tilde{m} = (m, r)$, is C-SUF-CMAA-secure if the CDH problem is hard.

Proof. Suppose that there exists an adversary A who forges a converted signature for the t-ABS scheme on a selected message $\tilde{\mu} = (\mu, \rho)$ and verification attribute set β in a chosen message and attribute set attack with probability ϵ. Then, one of the following is true:

either with probability at least $\epsilon/2$, A forges on a message (μ, ρ) such that for all queried messages (m, r) during the attack $g_1^m g_3^r \neq g_1^\mu g_3^\rho$,

or with probability at least $\epsilon/2$, A forges on a message (μ, ρ) such that for some queried message (m, r) during the attack $g_1^m g_3^r = g_1^\mu g_3^\rho$.

Let's call the above types of forgery as type 1 and type 2 forgery, respectively. We show that both types of forgery enables us to solve the CDH problem.

Dealing with A Type 1 Forger: Suppose we are given a type 1 forger A. We show how construct an algorithm B to solve the CDH problem. B is given (g, g^a, g^b) as input and calculates g^{ab} by running A as a subroutine and simulating the attack environment for it. When B initiates A, A outputs its target verification attribute set β and target message $\tilde{\mu} = (\mu, \rho)$. B sets $g_1 = g^a$ and $g_2 = g^b$. Then it chooses a random n degree polynomial $f(x)$ and another random n degree polynomial $u(x)$ such that $u(x) = -x^n$ if and only if $x \in \beta$. B then sets $t_i = g_2^{u(i)} g^{f(i)}$ for $i = 1, \cdots, n+1$. Note that we implicitly have $T(x) = g_2^{x^n + u(x)} g^{f(x)}$ since

$$\begin{aligned} T(x) &= g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)} = g_2^{x^n} \prod_{i=1}^{n+1} (g_2^{u(i)} g^{f(i)})^{\Delta_{i,N}(x)} \\ &= g_2^{x^n} g_2^{\sum_{i=1}^{n+1} u(i)\Delta_{i,N}(x)} g^{\sum_{i=1}^{n+1} f(i)\Delta_{i,N}(x)} \\ &= g_2^{x^n + u(x)} g^{f(x)} \end{aligned}$$

B then chooses z and γ randomly in $\mathbb{Z}_p$ and sets $g_3 = g_1^z$ and $h = g_1^{-\mu} g_3^{-\rho} \cdot g^\gamma$. It then gives $pk = (g, g_1, g_2, g_3, t_1, t_2, \ldots, t_{n+1}, h)$ to A who will start its secret key and signature queries. B responds to these queries as follows.

On a secret key query for a user with attribute set α, where $|\beta \cap \alpha| < d$, B defines $\Gamma = \beta \cap \alpha$ and lets Γ' be any set such that $\Gamma \subseteq \Gamma' \subseteq \alpha$ and $|\Gamma'| = d - 1$. B then sets $S = \Gamma' \cup \{0\}$ and calculates the following:

- for $i \in \Gamma'$ chooses r_i and λ_i randomly in $\mathbb{Z}_p$ and sets

$$ssk_i = \left\langle g_2^{\lambda_i} T(i)^{r_i}, g^{r_i} \right\rangle.$$

- for $i \in \alpha \setminus \Gamma'$ chooses r'_i randomly in $\mathbb{Z}_p$ and sets

$$ssk_{1i} = \left(g_1^{\frac{-f(i)}{i^n + u(i)}} \left(g_2^{i^n + u(i)} g^{f(i)} \right)^{r'_i} \right)^{\Delta_{0,S}(i)} \prod_{j \in \Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)}$$

$$ssk_{2i} = \left(g_1^{\frac{-1}{i^n + u(i)}} g^{r'_i} \right)^{\Delta_{0,S}(i)} .$$

The simulation is obviously correct for $i \in \Gamma'$. To see why the simulation for $i \in \alpha \setminus \Gamma'$ is also correct, let $q(x)$ be an $n - 1$ degree polynomial such that $q(i) = \lambda_i$ for $i \in \Gamma'$ and $q(0) = a$. This polynomial is implicitly chosen randomly by random choices of elements λ_i by B. Also note that $i^n + u(i) \neq 0$ for all $i \notin \beta$, which covers all elements $i \in \alpha \setminus \Gamma'$. Furthermore, let's implicitly set $r_i = \left(r'_i - \frac{a}{i^n+u(i)}\right)\Delta_{0,S}(i)$ and we will have:

$$\begin{aligned}
ssk_{1i} &= \left(g_1^{\frac{-f(i)}{i^n+u(i)}}\left(g_2^{i^n+u(i)}g^{f(i)}\right)^{r'_i}\right)^{\Delta_{0,S}(i)}\prod_{j\in\Gamma'}g_2^{\lambda_j\Delta_{j,S}(i)}\\
&= \left(g^{\frac{-af(i)}{i^n+u(i)}}\left(g_2^{i^n+u(i)}g^{f(i)}\right)^{r'_i}\right)^{\Delta_{0,S}(i)}\prod_{j\in\Gamma'}g_2^{\lambda_j\Delta_{j,S}(i)}\\
&= \left(g_2^a\left(g_2^{i^n+u(i)}g^{f(i)}\right)^{\frac{-a}{i^n+u(i)}}\left(g_2^{i^n+u(i)}g^{f(i)}\right)^{r'_i}\right)^{\Delta_{0,S}(i)}\prod_{j\in\Gamma'}g_2^{\lambda_j\Delta_{j,S}(i)}\\
&= \left(g_2^a\left(g_2^{i^n+u(i)}g^{f(i)}\right)^{r'_i-\frac{-a}{i^n+u(i)}}\right)^{\Delta_{0,S}(i)}\prod_{j\in\Gamma'}g_2^{\lambda_j\Delta_{j,S}(i)}\\
&= g_2^{a\Delta_{0,S}(i)}T(i)^{r_i}\prod_{j\in\Gamma'}g_2^{\lambda_j\Delta_{j,S}(i)}\\
&= g_2^{q(0)\Delta_{0,S}(i)+\sum_{j\in\Gamma'}q(j)\Delta_{j,S}(i)}T(i)^{r_i}\\
&= g_2^{q(i)}T(i)^{r_i}
\end{aligned}$$

and

$$ssk_{2i} = \left(g_1^{\frac{-1}{i^n+u(i)}}g^{r'_i}\right)^{\Delta_{0,S}(i)} = \left(g^{r'_i-\frac{a}{i^n+u(i)}}\right)^{\Delta_{0,S}(i)} = g^{r_i}.$$

Also note that random choices of elements r'_i implies random elements r_i. Thus, ssk is simulated correctly by B.

On a signature query by a user with attribute set α on a message (m, r), where $(m, r) \neq (\mu, \rho)$ or $|\beta\cap\alpha| < d$, B simulates the signature as follows:

- If $|\beta \cap \alpha| < d$ then B calculates ssk as above and then calculates the signature as follows for random choices of s_i

$$\sigma = \left\langle\{ssk_{1i}\left(g_1^m g_3^r \cdot h\right)^{s_i}, ssk_{2i}, g^{s_i}\}_{i\in\alpha}\right\rangle.$$

- Otherwise B chooses a random $d-1$ degree polynomial $q'(x)$ such that $q'(0) = \frac{\gamma}{m-\mu+z(r-\rho)}$ and random s'_i and r_i in $\mathbb{Z}_p$ for all $i \in \alpha$ and sets

$$\sigma = \left\langle\left\{g_2^{q'(i)}T(i)^{r_i}\left(g_1^m g_3^r \cdot h\right)^{s'_i}, g^{r_i}, g_2^{\frac{-1}{m-\mu+z(r-\rho)}}g^{s'_i}\right\}_{i\in\alpha}\right\rangle.$$

In the former case, B's simulation is correct for obvious reasons. We show that the simulation is correct for the latter case as well. Note that since A is a type 1 forger, we have $g_1^m g_3^r \neq g_1^\mu g_3^\rho$, which implies that $m + zr \neq \mu + z\rho$ (since $g_3 = g_1^z$). This ensures that $m - \mu + z(r - \rho) \neq 0$ and the above assignments are well defined. Now, let's (implicitly) define $s_i = s'_i - \frac{b}{m-\mu+z(r-\rho)}$ and $q(i) = q'(i) + a + \frac{\gamma}{m-\mu+z(r-\rho)}$. We have:

$$\begin{aligned}
\sigma_{1i} &= g_2^{q'(i)} T(i)^{r_i} \left(g_1^m g_3^r \cdot h\right)^{s'_i} \\
&= g_2^{q'(i)} T(i)^{r_i} \left(g_1^m g_3^r \cdot h\right)^{\frac{b}{m-\mu+z(r-\rho)}} \left(g_1^m g_3^r \cdot h\right)^{s'_i - \frac{b}{m-\mu+z(r-\rho)}} \\
&= g_2^{q'(i)} T(i)^{r_i} \left(g_1^m g_3^r \cdot g_1^{-\mu} g_3^{-\rho} g^{\gamma}\right)^{\frac{b}{m-\mu+z(r-\rho)}} \left(g_1^m g_3^r \cdot h\right)^{s_i} \\
&= g_2^{q'(i)} T(i)^{r_i} \left(g_1^{m-\mu+z(r-\rho)} \cdot g^{\gamma}\right)^{\frac{b}{m-\mu+z(r-\rho)}} \left(g_1^m g_3^r \cdot h\right)^{s_i} \\
&= g_2^{q'(i)} T(i)^{r_i} g_1^b \cdot g^{\frac{b\gamma}{m-\mu+z(r-\rho)}} \left(g_1^m g_3^r \cdot h\right)^{s_i} \\
&= g_2^{q'(i)} T(i)^{r_i} g_2^a \cdot g_2^{\frac{\gamma}{m-\mu+z(r-\rho)}} \left(g_1^m g_3^r \cdot h\right)^{s_i} \\
&= g_2^{q(i)} T(i)^{r_i} \left(g_1^m g_3^r \cdot h\right)^{s_i}
\end{aligned}$$

and

$$\sigma_3 = g_2^{\frac{-1}{m-\mu+z(r-\rho)}} g^{s'_i} = g^{s'_i - \frac{b}{m-\mu+z(r-\rho)}} = g^{s_i}$$

Note that random choices of s'_i and $q'(x)$ imply randomness of s_i and $q(x)$. We also have $q(0) = q'(0) + a + \frac{\gamma}{m-\mu+z(r-\rho)} = a$. Thus B's simulations in the latter case are correct as well.

Finally, the adversary A outputs a forgery $\tilde{\sigma}$ for the verification attribute set β and message (μ, ρ). Since $\tilde{\sigma}$ is a valid converted signature, we have:

$$\prod_{i \in B} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^{\mu} g_3^{\rho} \cdot h, \sigma_{3i})} \right)^{\Delta_{i,B}(0)} = e(g_2, g_1).$$

Now, note that $g_1^{\mu} g_3^{\rho} \cdot h = g^{\gamma}$ and also for any $i \in \beta : T(i) = g^{f(i)}$. Thus the above equation can be rewritten as follows:

$$\prod_{i \in B} \left(\frac{e(\sigma_{1i}, g)}{e(\sigma_{2i}^{f(i)}, g) \cdot e(\sigma_{3i}^{\gamma}, g)} \right)^{\Delta_{i,B}(0)} = e(g^{ab}, g).$$

Hence B is able to calculate g^{ab} and solve the CDH problem instance as follows:

$$\prod_{i \in B} \left(\frac{\sigma_{1i}}{\sigma_{2i}^{f(i)} \cdot \sigma_{3i}^{\gamma}} \right)^{\Delta_{i,B}(0)} = g^{ab}.$$

Dealing with A Type 2 Forger: Suppose we are given a type 2 forger A. We show how construct an algorithm C to solve the discrete logarithm and hence the CDH problem. C is given (g, g^z) as input and calculates z by running A as a subroutine and simulating the attack environment for it. When C initiates A, A outputs its target attribute set β and target message $\tilde{\mu} = (\mu, \rho)$. C follows the Setup algorithm of the t-ABS scheme to generate msk and mpk except for g_3 which it sets as $g_3 = g^z$. Thus, C picks y randomly from $\mathbb{Z}_p$ and sets $g_1 = g^y$ and also chooses random elements g_2, h, t_1, t_2, ..., t_{n+1} from $\mathbb{G}_1$. $T(x)$ is defined in the same way. C then keeps $msk = y$ private and gives $mpk = (g, g_1, g_2, g_3, t_1, t_2, \ldots, t_{n+1}, h)$ to the forger. Since C knows msk, it is able to answer A's secret key and signature queries by simply running the KeyGen and Sign algorithms of the t-ABS scheme.

Finally, the adversary A outputs a forgery $\tilde{\sigma}$ for the verification attribute set β and message (μ, ρ). Since A is a type 2 forger, for some queried message (m, r) during the attack we have $g_1^m g_3^r = g_1^{\mu} g_3^{\rho}$, hence $ym + zr = y\mu + z\rho$. Thus C can calculate the discrete logarithm as $z = y\frac{\mu - m}{r - \rho}$. Note that $r - \rho \neq 0$, since otherwise $r = \rho$ together with $ym + zr = y\mu + zr$ would imply that $m = \mu$, hence $(\mu, \rho) = (m, r)$ and (μ, ρ) would not count as a forgery. ■

D The Concrete t-ABCS Scheme

$\mathsf{Setup}(1^k)$: Pick y randomly from $\mathbb{Z}_p$ and set $g_1 = g^y$. Pick random elements g_2, g_3, h, t_1, t_2, ..., t_{n+1} from $\mathbb{G}_1$. Define

$$T(x) \triangleq g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)}$$

and output the following:

$$msk = y \quad \text{and} \quad mpk = (g, g_1, g_2, g_3, t_1, t_2, \ldots, t_{n+1}, h)$$

$\mathsf{KeyGen}(msk, A)$: Choose a random $d-1$ degree polynomial $q(x)$ such that $q(0) = y$, choose random elements r_i in $\mathbb{Z}_p$ for $i \in A$, and output

$$ssk = \Big\langle \quad \Big\{ \quad g_2^{q(i)} T(i)^{r_i}, \quad g^{r_i} \quad \Big\}_{i \in A} \quad \Big\rangle$$

$\mathsf{Sign}(ssk, (m, r))$: Parse the signing key as $ssk = \big\langle \{ssk_{1i}, ssk_{2i}\}_{i \in A} \big\rangle$, pick random elements s_i in $\mathbb{Z}_p$ for all $i \in A$, and output

$$\sigma = \Big\langle \quad A, \quad \Big\{ \quad ssk_{1i}\,(g_1^m g_3^r \cdot h)^{s_i}, \quad ssk_{2i}, \quad g^{s_i} \quad \Big\}_{i \in A} \quad \Big\rangle$$

$\mathsf{Verify}(mpk, (m, r), \tilde{\sigma}, B)$: Parse the signature as $\sigma = \big\langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \big\rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$ and check if the following equation holds:

$$\prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m g_3^r \cdot h, \sigma_{3i})} \right)^{\Delta_{i,S}(0)} = e(g_2, g_1)$$

$\mathsf{Convert}(mpk, (m, r), \sigma, B)$: Parse the signature as $\sigma = \big\langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \big\rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$. Calculate the converted signature components as follows:

$$\begin{array}{llll} \text{for all } i \in S: & \tilde{\sigma}_{1i} \leftarrow \sigma_{1i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow \sigma_{2i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow \sigma_{3i}^{1/\Delta_{i,B \setminus S}(0)} \\ \text{for all } i \in B \setminus S: & \tilde{\sigma}_{1i} \leftarrow (T(i) g_1^m g_3^r h)^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} \end{array}$$

and output $\tilde{\sigma} = \big\langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \big\rangle$.

$\mathsf{CvtVerify}(mpk, (m, r), \tilde{\sigma}, B)$: Parse the converted signature as $\tilde{\sigma} = \big\langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \big\rangle$ and check if the following equation holds:

$$\prod_{i \in B} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(g_1^m g_3^r \cdot h, \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} = e(g_2, g_1) \tag{3}$$

E Security of Our Concrete Additional Protocols

Theorem 2. Our t-ABS$^+$ scheme achieves both weak and full signer-attribute privacy.

Proof. It is easy to see that the converted signature cannot reveal more than the d selected attributes, since it simply does not include any components corresponding to other attributes. The proof of our iVerify protocol being a zero knowledge proof of knowledge of a converted signature is a simpler version of the proof of the same properties for iHVerify and iCVerify protocols. The latter proof is part of the proof of Theorem 4 which comes next. We omit the proof to avoid repetition. ■

The protocol iCSign is correct for obvious reasons. Correctness of the iHVerify and iCVerify protocols is also easy to see, considering the fact that the equation $\prod_{i\in B} u_{1i}^{\tau_i} = u_0 u_2 u_{31}^m u_{32}^r u_{33}$ is just a rearrangement of Equation 3. Now, we show that the protocols are secure as per our definitions.

Theorem 4. Our protocols iCSign, iHVerify, and iCVerify are secure for the user, signer, prover and verifier.

Proof. For the iCSign protocol, the only information the signer receives about m during the protocol is the commitment M, since the first step of the protocol is zero knowledge. Now since M perfectly hides m, the protocol satisfies the "security for the user" property. Technically, a distinguisher for the protocol implies either a distinguisher for the zero knowledge protocol in the first step or a distinguisher for the commitment scheme, which in turn contradicts witness indistinguishability of the zero knowledge protocol or the hiding property of the commitment scheme, respectively.

To prove the second property for iCSign protocol, we introduce the following simulator. The simulator first runs the knowledge extractor for the proof of knowledge in the first step of the protocol, which gives a pair (m, r) such that $M = g_1^m \cdot g_3^r$. Then the simulator queries a signature on (m, r) from the signing oracle and sends the acquired signature to the user. This shows that the "security for the signer" property is also satisfied.

To prove the zero knowledge property for the iHVerify and iCVerify protocols, note that the tuple that the prover sends to the verifier before performing the ZK-PoK, i.e., $\langle \{\hat{\sigma}_{1i}, \breve{\sigma}_{2i}, \breve{\sigma}_{3i}\}_{i\in B}\rangle$, consists of three random values independent of the actual converted signature. The reason is that these components are randomized by independent random elements τ_i, r_i', and s_i', respectively. Thus the following simulator is able to simulate the view of the verifier. First, the simulator picks random elements $\hat{\sigma}_{1i}$, $\breve{\sigma}_{2i}$, and $\breve{\sigma}_{3i}$ and sends them to the verifier. Then, it runs the simulator for the zero knowledge protocol in which simulates the rest of the verifier's view. Thus both the iHVerify and iCVerify protocols satisfy the "security for the prover" property.

To prove the proof of knowledge property for the iHVerify and iCVerify protocols, we must introduce an extractor for each that extracts a message converted-signature pair. Consider the following algorithm. First, the extractor receives the values $\langle \{\hat{\sigma}_{1i}, \breve{\sigma}_{2i}, \breve{\sigma}_{3i}\}_{i\in B}\rangle$ from the prover. Then it runs the extractor for the proof of knowledge in the last step of the protocol to extract a tuple $(m, r, \{\tau_i\}_{i\in B})$ if the protocol is iHVerify or $(m, r, r', \{\tau_i\}_{i\in B})$ if the protocol is iCVerify. Now, by letting $\breve{\sigma}_{1i} = \hat{\sigma}_{1i}^{\tau_i}$ the extractor gets a randomized converted signature $\breve{\sigma} = \langle \{\breve{\sigma}_{1i}, \breve{\sigma}_{2i}, \breve{\sigma}_{3i}\}_{i\in B}\rangle$, which is itself a valid converted signature. Thus, the "security for the verifier" property is satisfied for both protocols. ■

F A Standard-Model Existentially-Unforgeable t-ABS$^+$ Scheme

Signer attributes are assumed to be sets of at most n elements of $\mathbb{Z}_p$. Although generally, identities can be sets of at most n arbitrary strings and a collision resistant hash function is used to map the strings to elements of $\mathbb{Z}_p$. We use $N = \{1, 2, \ldots, n+1\}$ to denote the set of possible attributes. Let $\mathbb{G}_1 = \langle g\rangle$ be a group of prime order p and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be defined. Let $(e, \mathbb{G}_1, \mathbb{G}_2)$ be of public knowledge. We present the scheme for signing messages with (maximum) bit length ℓ. Although, the message space can be expanded to contain arbitrary messages using a collision resistant hash function to map strings to the set of binary strings with (maximum) length ℓ. Waters function $\mathrm{W}(\cdot)$ is defined as $\mathrm{W}(m) = h \prod h_i^{m_i}$, where m_i denotes the i-th bit of m.

Setup(1^k): Pick y randomly from $\mathbb{Z}_p$ and set $g_1 = g^y$. Pick random elements g_2, h, t_1, t_2, ..., t_{n+1}, h_1, h_2, ..., h_ℓ from $\mathbb{G}_1$. Define and output the following:

$$T(x) \triangleq g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)} \quad \text{and} \quad msk = y \quad \text{and} \quad mpk = (g, g_1, g_2, t_1, t_2, \ldots, t_{n+1}, h)$$

KeyGen(msk, A): Choose a random $d-1$ degree polynomial $q(x)$ such that $q(0)=y$, choose random elements r_i in $\mathbb{Z}_p$ for $i \in A$, and output

$$ssk = \Big\langle \quad \Big\{ \quad g_2^{q(i)} T(i)^{r_i}, \quad g^{r_i} \quad \Big\}_{i \in A} \quad \Big\rangle$$

Sign(ssk, m): Parse the signing key as $ssk = \langle \{ssk_{1i}, ssk_{2i}\}_{i\in A} \rangle$, pick random elements s_i in $\mathbb{Z}_p$ for all $i \in A$, and output

$$\sigma = \Big\langle \quad A, \quad \{ \quad ssk_{1i} \mathrm{W}(m)^{s_i}, \quad ssk_{2i}, \quad g^{s_i} \quad \}_{i \in A} \quad \Big\rangle$$

Verify(mpk, m, σ, B): Parse the signature as $\sigma = \langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i\in A} \rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$ and check if the following equation holds:

$$\prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(\mathrm{W}(m), \sigma_{3i})} \right)^{\Delta_{i,S}(0)} = e(g_2, g_1)$$

The concrete conversion and converted-signature verification algorithms are as follows:

Convert(mpk, m, σ, B): Parse the signature as $\sigma = \langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i\in A} \rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$. Calculate the converted signature components as follows:

$$\begin{array}{llll} \text{for all } i \in S: & \tilde{\sigma}_{1i} \leftarrow \sigma_{1i}^{1/\Delta_{i,B\setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow \sigma_{2i}^{1/\Delta_{i,B\setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow \sigma_{3i}^{1/\Delta_{i,B\setminus S}(0)} \\ \text{for all } i \in B \setminus S: & \tilde{\sigma}_{1i} \leftarrow (T(i)\mathrm{W}(m))^{1/\Delta_{i,B\setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow g^{1/\Delta_{i,B\setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow g^{1/\Delta_{i,B\setminus S}(0)} \end{array}$$

and output $\tilde{\sigma} = \langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i\in B} \rangle$.

CvtVerify$(mpk, m, \tilde{\sigma}, B)$: Parse the converted signature as $\tilde{\sigma} = \langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i\in B} \rangle$. Check if the following equation holds:

$$\prod_{i \in B} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(\mathrm{W}(m), \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} = e(g_2, g_1)$$

Furthermore, the iVerify protocol flow is as follows:

1. The signature holder randomizes the converted signature by first choosing random elements s_i' and r_i' for $i \in B$ and then calculating the following. Note that the resulting randomized converted signature is a valid converted signature itself.

$$\breve{\sigma}_{1i} = \tilde{\sigma}_{1i} \cdot T(i)^{r_i'} \mathrm{W}(m)^{s_i'} \quad \text{and} \quad \breve{\sigma}_{2i} = \tilde{\sigma}_{2i} \cdot g^{r_i'} \quad \text{and} \quad \breve{\sigma}_{3i} = \tilde{\sigma}_{3i} \cdot g^{s_i'}$$

2. The signature holder chooses random values τ_i for all $i \in B$ and sets $\hat{\sigma}_{1i} \leftarrow \breve{\sigma}_{1i}^{1/\tau_i}$ and sends $\langle \{\hat{\sigma}_{1i}, \breve{\sigma}_{2i}, \breve{\sigma}_{3i}\}_{i\in B} \rangle$ to the verifier.

3. Both the signature holder and the verifier calculate the following for all $i \in B$:

$$\begin{array}{ll} u_0 \leftarrow e(g_2, g_1) & u_{1i} \leftarrow e(\hat{\sigma}_{1i}, g)^{\Delta_{i,B}(0)} \\ u_2 \leftarrow \prod_{i\in B} e(T(i), \breve{\sigma}_{2i})^{\Delta_{i,B}(0)} & u_3 \leftarrow \prod_{i \in B} e(\mathrm{W}(m), \breve{\sigma}_{3i})^{\Delta_{i,B}(0)} \end{array}$$

4. The signature holder performs the following ZK-PoK for the verifier:

$$\text{ZK-PoK}\{ \quad (\{\tau_i\}_{i\in B}) : \quad \prod_{i\in B} u_{1i}^{\tau_i} = u_0 u_2 u_3 \quad \}$$

G Security of Our Concrete t-ABACS System

We prove the following theorem.

Theorem 5 Our concrete t-ABACS system emulates the ideal model for a t-ABACS if the underlying t-ABCS scheme is EUF-CMAA-secure.

Proof sketch. Let E be the environment and A_R be the adversary in the real model. To prove that our system emulates the ideal model, we must present a ideal-model adversary A_I such that E cannot distinguish if it is talking to the ideal system (of ideal parties and A_I) or to the real system (of real parties and A_R). To do this, we run a copy of A_R and *translate* its communications into the ideal model. As for A_R's communication with E, no translation is needed. However, we must be able to do the two following translations: (i) translate any message from T, intended for an ideal corrupted party, to a message that A_R understands, i.e., a message intended for a real corrupted party, and (ii) translate any message from A_R (i.e., from a real corrupted party), intended for a real honest party, to a message that T understands. We introduce a translator S that takes care of the above in the following. The combination of S and A_R can be seen as the ideal-model adversary A_I that we propose. A depiction of our simulation of the ideal model using the real adversary A_R and the translator S comes is Figure 2.

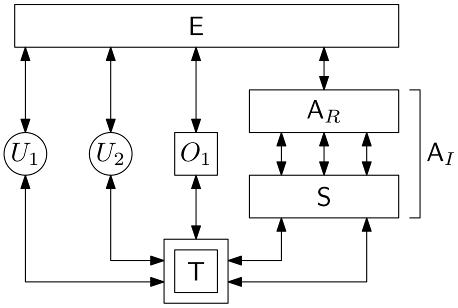

Figure 2: Simulation of the ideal model in Figure 1 (left) in the proof using the real adversary A_R from the same figure (right) as a black box with its three incoming/outgoing communication channels. S translates these communications into communications with T.

Since all the protocols in our systems are between a user and an organization, S needs to translate messages in two cases: either a protocol between a corrupted user and an honest organization or a protocol between an honest user and a corrupted organization. The translator S first generates secret signing keys for the ideal organizations and then does the following based on the relevant case. In the following we assume that each pseudonym N corresponds with only one pair (SK, r), since otherwise, the binding property of the commitment scheme is broken. Technically speaking, we are able to use any adversary that breaks the binding property to solve the discrete logarithm problem. Thus, we exclude such adversaries from the following simulation.

Case 1. On the event that a corrupted user initiates a protocol with an honest organization, S does as follows based on the initiated protocol:

- When a corrupted user initiates a real-model $\mathsf{FormNym}$ protocol with an honest organization O on pseudonym N, S runs the extractor for the proof of knowledge of a committed value protocol and extracts the user's secret key SK and the corresponding r. If SK is a new key, S sets up login information (U, K_U) with T for the corrupted user and stores (U, K_U) under the new entry SK in its records. Otherwise, S looks SK up in its records and retrieves the login information (U, K_U). Then S runs the ideal-model $\mathsf{FormNym}$ protocol with T on N. Upon completion of the protocol, S forwards the output to the corrupted user and finishes the real-model $\mathsf{FormNym}$ protocol execution. Finally, S stores (O, N, r) under SK in its records. Note that this record corresponds to $(U, O, N) \in L_{\text{N}}$ in T's records.

- When a corrupted user initiates a real-model GrantCred protocol with an honest organization I on pseudonym N, S runs the simulator for the iCSign protocol. This simulates the interaction for the user. At some point, the simulator queries for a signature on some message (SK, r). S stops the simulator execution at this time and looks SK up in its records. Then it retrieves the corresponding login information (U, K_U). Using this login information, S starts an ideal-model GrantCred protocol with T on N. When T informs S that the credential is granted, S computes the queried signature σ on (SK, r) using the secret signing key it has generated before for I. Then it answers the signature query of the simulator and resumes execution of the simulator code. Finally, S updates its record of (I, N, r) to (I, N, r, σ). Note that this record corresponds to $(N, A_I) \in L_{\mathrm{C}}$ in T's records. Also note that the successful completion of the ideal-model GrantCred protocol means that $(U, I, N) \in L_{\mathrm{N}}$ and hence (I, N, r) exists in records under SK.

- When a corrupted user initiates a real-model VerifyCred protocol with an honest organization V, S runs the extractor for the iHVerify protocol and extracts the corrupted user's secret key SK, the corresponding r, and a converted signature $\tilde{\sigma}$ on (SK, r). S proceeds only if $\tilde{\sigma}$ is a valid converted signature on (SK, r) with respect to B_V. S then looks SK up in its records and retrieves the corresponding (U, K_U). It also searches for r in its records on SK and checks if it finds a matching record (I, N', r, σ). If no such matching record is found S checks if any of the corrupted organizations have at least d attributes in common with B_V. If such an organization I is found, S first runs the ideal-model FormNym and GrantCred protocols on behalf of both U and I with T to form a pseudonym N' and get a credential on it. If neither a matching record nor a suitable corrupted organization is found, S outputs $((SK, r), \tilde{\sigma})$ and halts. Then S runs an ideal-model VerifyCred protocol with T on issuer pseudonym N' and relays the outcome of the protocol back to the corrupted user.

- When a corrupted user initiates a real-model VerifyCredOnNym protocol with an honest organization V on pseudonym N, S runs the extractor for the iCVerify protocol and extracts the corrupted user's secret key SK, the corresponding r and r_V, and a converted signature $\tilde{\sigma}$ on (SK, r). S proceeds only if $\tilde{\sigma}$ is a valid converted signature on (SK, r) with respect to B_V. S then looks SK up in its records and retrieves the corresponding (U, K_U). It also searches for r in its records on SK and checks if it finds a matching record (I, N', r, σ). If no such matching record is found or the converted signature is not equal to $\tilde{\sigma}$, S checks if any of the corrupted organizations have at least d attributes in common with B_V. If such an organization I is found, S first runs the ideal-model FormNym and GrantCred protocols on behalf of both U and I with T to form a pseudonym N' and get a credential on it. If neither a matching record nor a suitable corrupted organization is found, S outputs $((SK, r), \tilde{\sigma})$ and halts. Then S runs an ideal-model VerifyCredOnNym protocol with T on issuer pseudonym N' and verifier pseudonym N and relays the outcome of the protocol back to the corrupted user.

Case 2. On the event that an honest user initiates a protocol with a corrupted organization, T contacts S. S does as follows based on the initiated protocol:

- When an honest user initiates an ideal-model FormNym protocol with a corrupted organization O on pseudonym N, T contacts S with a pseudonym establishment request on N. S picks a random secret key SK' for this user. Then S calculates a commitment N' on SK' with a random r' and runs the real-model FormNym protocol as a user with O. Upon successful completion of the protocol, S responds to T to complete the ideal FormNym protocol and establish the pseudonym N with the honest user. Finally, S stores (O, N, r', N') under secret key SK' in its records. Note that this record corresponds to a record $(U, O, N) \in L_{\mathrm{N}}$ in T's records for some U that we do not know.

- When an honest user initiates an ideal-model GrantCred protocol with a corrupted organization I on pseudonym N, T contacts S with a credential issuance request on N if N is the honest user's pseudonym with I (i.e., $(U, I, N) \in L_{\mathrm{N}}$ for U being the honest user). S first searches its records for a pseudonym N and retrieves the corresponding SK' and (I, N, r', N'). Such a record exists since $(U, I, N) \in L_{\mathrm{N}}$. Then it runs the iCSign protocol as a user with I on input (SK', r') and pseudonym N'. Upon completion of the protocol, if S gets a valid signature σ' on (SK', r'), it notifies T that a credential is granted and completes the ideal-model GrantCred protocol with T. Finally, S updates the record (I, N, r', N') to (I, N, r', N', σ'). Note that this record corresponds to $(N, A_I) \in L_{\mathrm{C}}$ in T's records.

- When an honest user initiates an ideal-model $\mathsf{VerifyCred}$ protocol with a corrupted organization V on some issuer pseudonym, T contacts S with a credential verification request if the user has a credential on its issuer pseudonym, from an issuer with at least d common attributes with B_V. S runs the simulator for the $\mathsf{iCVerify}$ protocol. This simulates the interaction with V. Upon completion of the simulation, S acknowledges credential verification to T and finishes the ideal-model $\mathsf{VerifyCred}$ protocol execution.
- When an honest user initiates an ideal-model $\mathsf{VerifyCredOnNym}$ protocol with a corrupted organization V on some issuer pseudonym and a verifier pseudonym N, T contacts S with a credential verification request on N if the user has a credential on its issuer pseudonym, from an issuer with at least d common attributes with B_V. S searches its records for a pseudonym N and retrieves the corresponding SK' and (V, N, r', N'). Then S runs the simulator for the $\mathsf{iCVerify}$ protocol on N'. This simulates the interaction with V. Upon completion of the simulation, S acknowledges credential verification to T and finishes the ideal-model $\mathsf{VerifyCredOnNym}$ protocol execution.

The simulation only halts when a corrupted user manages to produce a pair $((SK, r), \tilde{\sigma})$ such that $\tilde{\sigma}$ is a valid converted signature on (SK, r) and is not issued through a $\mathsf{GrantCred}$ protocol by an honest organization. This means that the signature basically is a forgery! Thus if a combination of an environment E and an adversary A_I (itself made of S and A_R) causes the simulation to halt, we can use them to construct an algorithm that forges signatures for the t-ABCS scheme. In the following we show that if the simulation does not halt, then it will provide an indistinguishable simulation.

Let's call each of the above protocols a *simulated* protocol and the above system containing the translator S the *simulated* system. Assume that the environment E schedules a total of ν protocols. We define an *i-th hybrid* system for $i \in \{0, 1, \ldots, \nu\}$ as a system in which the first i protocols are simulated protocols and the rest of the protocols are ideal protocols. With such a definition, the ideal system can be seen as the zeroth hybrid and the simulated system as the ν-th hybrid.

Now, if a polynomial time environment E distinguishes between the ideal system and the simulated system, then it should be able to distinguish between $(j-1)$-th and j-th hybrids for some $j \in \{1, \ldots, \nu\}$. Since the only difference between the two hybrids is the j-th protocol, E should be able to distinguish between a simulated j-th protocol and an ideal j-th protocol. Depending on which protocol the j-th protocol is, we have the following cases:

- If the j-th protocol is a $\mathsf{FormNym}$ protocol between a corrupted user and an honest organization, then E should be able to distinguish between a knowledge extractor and a verifier in the proof of knowledge of a committed value protocol, which is a contradiction.
- If the j-th protocol is a $\mathsf{GrantCred}$ protocol between a corrupted user and an honest organization, then we use E to construct an algorithm that distinguishes between a simulator and a signer in the iCSign protocol.
- If the j-th protocol is a $\mathsf{VerifyCred}$ or a $\mathsf{VerifyCredOnNym}$ protocol between a corrupted user and an honest organization, then E should be able to distinguish between a knowledge extractor and a verifier in the $\mathsf{iHVerify}$ or $\mathsf{iCVerify}$ protocol, respectively, which is a contradiction.
- The j-th protocol cannot be a $\mathsf{FormNym}$ protocol between an honest user and a corrupted organization, since the simulation exactly follows the ideal and real-model protocol procedure.
- If the j-th protocol is a $\mathsf{GrantCred}$ protocol between an honest user and a corrupted organization, then we use E to construct an algorithm that distinguishes between interactions with users with different private inputs in the iCSign protocol.
- If the j-th protocol is a $\mathsf{VerifyCred}$ or a $\mathsf{VerifyCredOnNym}$ protocol between an honest user and a corrupted organization, then we use E to construct an algorithm that distinguishes between a prover

and a simulator in the iHVerify or iCVerify protocol, respectively.

Hence we have shown that unless either the t-ABS scheme is forgeable, the commitment scheme is not secure, or either of the iCSign, iHVerify, or iCVerify protocols is not secure in the sense of our definitions, then we succeed in simulating an indistinguishable system for E and this completes the proof. ■