Key Exchange with Anonymous Authentication using DAA-SIGMA Protocol

Jesse Walker and Jiangtao Li

Intel Labs
{jesse.walker, jiangtao.li}@intel.com

Abstract. Anonymous digital signatures such as Direct Anonymous Attestation (DAA) and group signatures have been a fundamental building block for anonymous entity authentication. In this paper, we show how to incorporate DAA schemes into a key exchange protocol between two entities to achieve anonymous authentication and to derive a shared key between them. We propose a modification to the SIGMA key exchange protocol used in the Internet Key Exchange (IKE) standards to support anonymous authentication using DAA. Our key exchange protocol can be also extended to support group signature schemes instead of DAA. We present a secure model for key exchange with anonymous authentication derived from the Canetti-Krawczyk key-exchange security model. We prove that our DAA-SIGMA protocol is secure under our security model.

1 Introduction

Anonymous digital signatures such as group signatures [22, 2, 5, 6], Direct Anonymous Attestation (DAA) [7, 8, 23, 24, 14], and anonymous credentials [15–17] play an important role in privacy enhanced technologies. They allow an entity (e.g., a user, a computer platform, or a hardware device) to create a signature without revealing its identity. Anonymous signatures also enable anonymous entity authentication.

The concept of group signature scheme was first introduced by Chaum and van Heyst [22] in 1991. In a group signature scheme, all the group members share a group public key, yet each member has a unique private key. A group signature created by a group member is anonymous to all the entities except a trusted group manager. Many group signature schemes have been proposed, e.g., in [2, 5, 6, 29]. Direct Anonymous Attestation (DAA) was first introduced by Brickell, Camenisch, and Chen [7] for remote anonymous authentication of a Trusted Platform Module (TPM). DAA can be seen as a special group signature scheme without the “open” feature. DAA has received a lot of attentions in the trusted computing community, and many DAA schemes have been developed recently, e.g., in [8, 23, 25, 24, 14].

Anonymous digital signatures have attracted a lot of industry attentions recently. For example, the Trusted Computing group (TCG), a global industrial standard body [35], adopted the original DAA scheme [7] and standardized it in the TCG TPM Specification v1.2 [34]. The same DAA scheme has recently been adopted by ISO/IEC as an international standard [1]. DAA has been implemented and already shipped in millions of TPMs. Intel has implemented an extension of DAA called Enhanced Privacy ID [10–12] in the Intel P55 Ibex Peak chipsets [13]. Recently, ISO/IEC starts to develop two new international standards: one on anonymous digital signatures including group signature schemes and DAA schemes, the other on anonymous entity authentication using anonymous digital signatures.

DAA can be used for anonymous authentication in a straight-forward manner: a verifier sends a challenge message including a nonce to a group member; the group member can authenticate to the verifier anonymously using his private key to create a DAA signature on the message. After verifying the DAA signature, the verifier is convinced that the group member is a valid DAA signer, but he does not learn who created the signature. TPM uses this method for anonymous authentication in order to obtain an attestation identity key credential from an issuer. This authentication method is limited because the group member did not verify the verifier's credential and there is no shared key derived from the authentication.

Consider the following scenario: a trusted computing platform wants to download some protected resources from an Internet server and may also upload some of its sensitive data to the server. The platform wants to authenticate the server before it sends the data out. In the same time the server wants to make sure the platform is indeed trusted, e.g., ensure that the platform can protect the server's resources. DAA can be used in this example if the platform wants to authenticate to the server anonymously. However the naïve authentication method in the previous paragraph does not work well in this scenario for the following reasons:

1. The platform and server need to authenticate to each other at the same time.
2. It is desired to have a session key derived after the mutual authentication so that the platform does not need to repeatedly authenticate to the server, when each time it wants to get some resources from the server.

A natural extension of DAA in anonymous authentication is to embed DAA in a Diffie-Hellman key exchange protocol, so that two entities can authenticate to each other and derive a shared session key for future communication. There have been several proposals to incorporate DAA into key exchange protocols. Balfe et al. [3] proposed anonymous authentication in peer-to-peer networks by embedding DAA with TLS and IPsec. Leung and Mitchell [32] introduced an anonymous authentication protocol based on DAA. Recently, Cesena et al. [20] proposed an anonymous authentication protocol based on TLS and DAA including a reference implementation.

1.1 Our Contribution

It is easy to design simple Diffie-Hellman (DH) based key exchange protocols, but it is easier to get them wrong. For example, many DH-based key exchange protocols are vulnerable to man-in-the-middle attacks or identity misbinding attacks [28]. Although there have been several proposals to use DAA in Diffie-Hellman key exchange protocols [3, 32, 20] in the literature, to the best of our knowledge, there is no formal security model provided and none of these protocols has formal security proof to prove the security of these protocols. This is the motivation of this paper. There are two contributions of this paper. We describe each contribution briefly as follows.

A SECURITY MODEL FOR KEY EXCHANGE WITH ANONYMOUS AUTHENTICATION. We give a rigorous treatment to anonymous authentication and introduce a new security model for key exchange with anonymous authentication. Our security model derives from the Canetti-Krawczyk key exchange security model [18, 19]. In the Canetti-Krawczyk security model, identity plays an important role in the proof of security. However in anonymous authentication, the identity of the entity who wants to be anonymous cannot be revealed in the key exchange. This creates a significant challenge in the definition of secure model and in the design of the key exchange protocol.

A SECURE KEY EXCHANGE PROTOCOL WITH ANONYMOUS AUTHENTICATION. We develop a new key exchange protocol with anonymous authentication based on DAA and the SIGMA family of key exchange protocols from IPsec [31] and the Internet Key Exchange (IKE) standards [30]. We call our protocol the DAA-SIGMA protocol. We present a formal security analysis of our DAA-SIGMA protocol based on our security model.

1.2 Related Work

Besides using DAA for anonymous authentication in key exchange protocols [3, 32, 20], other mechanisms of anonymous authentication have been proposed recently. Cui and Cao proposed an anonymous authentication and key exchange protocol based on ring signatures [26]. Viet et al. proposed a password-based anonymous authentication key exchange protocol [36]. Yang and Zhang improved Viet et al.'s scheme with another anonymous password-based key exchange protocol [37]. Chai et al. proposed an efficient password-based authentication scheme that preserves user privacy [21].

Besides the Canetti-Krawczyk key exchange model [18] for security analysis of key exchange protocols, other models and tools have been proposed in the literature. For example, Meadows used an automated protocol analyzer to study the security of key exchange protocols [33]. Datta et al. developed a symbolic logic analyzer for proving security properties of key exchange protocols [27]. Bellare and Rogaway formalized the security of key exchange protocols in the realistic setting of concurrent sessions [4]. In this paper, we choose to use the Canetti-Krawczyk key exchange model as the basis because this model is well established and has been used to analyze the security of the SIGMA key exchange protocol.

1.3 Organization of the Paper

Rest of this paper is organized as follows. We first introduce our notation and briefly review the DAA schemes and the SIGMA key exchange protocol in Section 2. We then propose the security model of key exchange protocol with anonymous authentication in Section 3. We present our DAA-SIGMA protocol in Section 4 and give the security proof in Section 5. We discuss some extensions of our DAA-SIGMA protocol in Section 6. We conclude our paper in Section 7.

2 Background and Building Blocks

In this section, we first review our notations and terminologies then briefly review the basic concepts of DAA. Next we review the SIGMA key exchange protocol.

2.1 Notations

We use the following notations in this paper. Let P and Q to be two entities of the key exchange protocol.

- $\text{MAC}_k(m)$: a message authentication code of m computed using the key k.
- $\text{SIG}_P(m)$: a signature of m created by the entity P.
- ID_P: an identity of the entity P.
- CERT_P: a public key certificate of the entity P.
- PRF: a pseudo-random function.

2.2 Review of DAA

In this section, we review the specification and security model of DAA proposed in [9]. The security model in [9] is simpler than the original DAA definition [7] and easier to understand the security properties of DAA. There are four types of players in a DAA scheme: an issuer $\mathcal{I}$, a TPM $\mathcal{M}_i$, a host $\mathcal{H}_i$ and a verifier $\mathcal{V}_j$. $\mathcal{M}_i$ and $\mathcal{H}_i$ form a platform in the trusted computing environment and share the role of a DAA signer $\mathcal{S}_i$. To simplify our notation, we treat the DAA signer as a single entity in the rest of this paper. A DAA scheme has following polynomial-algorithms or interactive protocols ($\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link}$):

Setup : On input of a security parameter 1^k, $\mathcal{I}$ uses this randomized algorithm to produce a pair ($\mathtt{gpk}$, $\mathtt{isk}$), where $\mathtt{isk}$ is the issuer's secret key, and $\mathtt{gpk}$ is the public key including the global public parameters.

Join : A signer $\mathcal{S}_i$ and the issuer $\mathcal{I}$ run an interactive join protocol. In the end of the protocol, $\mathcal{S}_i$ outputs a secret key $\mathtt{sk}_i$ and a membership credential $\mathtt{cre}_i$ issued by $\mathcal{I}$. We denote the $\mathtt{sk}_i$ and $\mathtt{cre}_i$ pair as the signing key of $\mathcal{S}_i$. Note that the value of $\mathtt{sk}_i$ is unknown to $\mathcal{I}$.

Sign : On input of $\mathtt{gpk}$, $\mathtt{sk}_i$, $\mathtt{cre}_i$, a basename $\mathtt{bsn}_j$, and a message m, $\mathcal{S}_i$ uses this algorithm to produce a signature σ on m. The basename $\mathtt{bsn}_j$ can be either the name string of the verifier $\mathcal{V}_j$ or a special symbol $\perp$ and it is used for controlling the linkability.

Verify : On input of $\mathtt{gpk}$, $\mathtt{bsn}_j$, m, a candidate signature σ on m, and a revocation list $\mathtt{RL}$, $\mathcal{V}_j$ uses this deterministic algorithm to determine whether σ is valid. The revocation is out of the scope of this paper.

Link : On input of two signatures σ_0 and σ_1, $\mathcal{V}_j$ uses this deterministic algorithm to return linked, unlinked, or invalid signatures.

The formal security definition of DAA can be found in [7, 9]. For completeness of this paper, we review the formal security model of DAA in [9] in Appendix A. Informally, a DAA scheme is secure if the following properties hold:

- *Correctness.* A signature created using a valid signing key can be verified by any verifiers correctly.
- *Anonymity.* An adversary who are not in possession of a secret key $\mathtt{sk}$ cannot learn the identity of the signer of signatures created using $\mathtt{sk}$.
- *Unforgeability.* An adversary cannot forge a valid signature if he does not have a signing key or his signing keys have been all revoked.
- *User-controlled linkability.* Signatures created by the same set of $\mathtt{sk}$, $\mathtt{cre}$, $\mathtt{bsn}$ can be linked, if $\mathtt{bsn} \neq \perp$. However, signatures cannot be linked if they are created using different $\mathtt{bsn}$ or if $\mathtt{bsn} = \perp$.

If a signature σ was created using $\mathtt{bsn} = \perp$, then we call σ a random based signature; otherwise, we call σ a name based signature. For the DAA-SIGMA protocol, we first focus on random based signatures. We shall discuss how to adopt name based signatures in Section 6.

Let I be a DAA issuer. Let P be a DAA signer who has a valid signing key issued by I. We use the following notations in the rest of the paper:

- ID_I: the identity of the issuer I. For simplicity, we assume I creates only one group public key $\mathtt{gpk}$. One can figure out the corresponding $\mathtt{gpk}$ from ID_I. If the issuer creates multiple group public keys, we can use ID_G to denote the identity of the DAA group.

– $\textsc{cert}_I$: a public key certificate of I issued by a certificate authority to certify the group public key $\texttt{gpk}$.
– $\textsc{daa-sig}_P(m)$: a DAA signature on message m created by the entity P. Let $\texttt{sk}_P$ be P's secret key and $\texttt{cre}_P$ be P's membership credential, then $\textsc{daa-sig}_P(m)$ denotes $\mathsf{Sign}(\texttt{gpk}, \texttt{sk}_P, \texttt{cre}_P, \texttt{bsn}, m)$ where $\texttt{bsn} = \perp$ if $\texttt{bsn}$ is not provided as input.

2.3 Review of SIGMA Key Exchange Protocol

We now review the SIGMA key exchange protocol. This key exchange protocol is one of the Internet Key-Exchange (IKE) protocols [30] used to establish secret shared keys for use in the Internet Protocol Security (IPsec) standards [31]. The SIGMA key exchange protocol uses Diffie-Hellman key exchange with "sign-and-mac" mechanism. It is presented as follows: Let P and Q be two entities of the key exchange protocol, where P is the protocol initiator who activates the session and Q is the intended peer of the session. Let G be a cyclic group of prime order q where the Decisional Diffie-Hellman (DDH) problem is hard and g be a generator of G. Let s be a unique session identifier chosen by P, Q, or both. The SIGMA protocol has the following messages.

$$\begin{aligned}
&\text{Message 1 } (P \rightarrow Q): \ s, g^x\\
&\text{Message 2 } (P \leftarrow Q): \ s, g^y, \textsc{id}_Q, \textsc{mac}_{k_1}(s, \textsc{id}_Q), \textsc{sig}_Q(s, g^x, g^y)\\
&\text{Message 3 } (P \rightarrow Q): \ s, \textsc{id}_P, \textsc{mac}_{k_1}(s, \textsc{id}_P), \textsc{sig}_P(s, g^y, g^x)
\end{aligned}$$

In the above protocol, P chooses a random integer x and computes the ephemeral DH public key g^x. Analogously, Q chooses a random integer y and compute its ephemeral DH public key g^y. By exchanging g^x and g^y, both entities can compute g^{xy}, but "man-in-the-middle" attackers cannot. Both entities then derive a session key k_0 and a $\textsc{mac}$ key k_0 from g^{xy}. In message 2, Q computes a $\textsc{mac}$ of its identity using k_1 and signs the ephemeral DH public keys (g^x, g^y) using its signing key. P can verify the signature to ensure that the message 2 indeed comes from entity Q and verify the $\textsc{mac}$ to ensure that Q knows k_1, thus knows the session key k_0 as well. Similarly, P computes a $\textsc{mac}$ of its identity and a signature of the DH public keys in message 3. After Q verifies the message 3, both entities have established a session key k_0 and can use k_0 for further communications.

3 Security Model

In this section, we first review the Canetti-Krawczyk key exchange model [18, 19], and then describe how to extend this key exchange model to support anonymous authentication.

3.1 Review of Canetti-Krawczyk Key Exchange Model

This section reviews the Canetti-Krawczyk model [18] for authenticated key establishment protocols. A *session key adversary*, or *SK-adversary*, $\mathcal{A}$ is a probabilistic polynomial time algorithm that controls all communication between any parties using the set of queries specified in Table 1 below. An SK-adversary accomplishes this control by issuing queries to a set of *oracles* $\mathcal{O}_{s,P}$. An oracle $\mathcal{O}_{s,P}$ represents the protocol instance identified by s initiated by principal P. This communications instance is called a session. Each oracle responds to the queries, and the outputs of all the oracles $\mathcal{O}_{s,P}$ collectively represent the history of the protocol. The SK-adversary receives all output, and the uses this output to decide which oracles to activate next.

Table 1. Adversarial Queries

Query	Description
Send(P, Q, s, m)	Oracle $\mathcal{O}_{s,P}$ responds to message m to Q according to the protocol specification and decides whether to accept, reject, or continue. Accept means the oracle has completed the protocol with partner Q, a session id s, session key K, and $\mathcal{O}_{s,P}$ signals its acceptance by including (P, s, Q) with its public output. Reject means the protocol failed. Continue means the oracle is waiting for its next protocol step. $\mathcal{O}_{s,P}$ returns its decision to the adversary A.
Session-Key-Reveal(P, Q, s)	If an oracle $\mathcal{O}_{s,P}$ has accepted and holds a session key K, the oracle returns K to SK-adversary $\mathcal{A}$. Otherwise $\mathcal{O}_{s,P}$ ignores this query. This query models a key leaking via break-ins, cryptanalysis, careless disposal of keys, and the like.
State-Reveal(P)	If the oracle $\mathcal{O}_{s,P}$ has accepted or holds a session key, it ignores this query. Otherwise the oracle returns its internal ephemeral session state, but no long-lived secrets. This query models the compromise of a single session.
Corrupt(P)	P responds to this query by yeielding its entire internal state, including all of the state held by all its oracles $\mathcal{O}_{s,P}$, and any long-lived keys. Subsequently the SK-adversary controls all of P's actions.
Expire-Session(P, Q, s)	This query causes oracle $\mathcal{O}_{s,P}$ to delete its session state, and oracle $\mathcal{O}_{s,P}$ will henceforth ignore any subsequent Session-Key-Reveal query, because it no longer has the targeted session key.
Test(P, Q, s)	An SK-adversary $\mathcal{A}$ may use the test query only once. This query is used to measure that adversary's efficacy at distinguishing a real session key from a randomly generated one. If $\mathcal{O}_{s,P}$ has accepted with a session key K, then it selects a bit value b randomly; if $b = 0$, then it returns K and otherwise generates a random value K, which it returns instead. If the oracle has not accepted or does not have the session key, it does not respond to the test query.

A session is called *exposed* if an SK-adversary issues a session-key-reveal query against its oracle or an state-reveal or corrupt query against one of the session's principals.

As noted, the model uses the test query to formalize the efficacy of an attack. The adversary may apply the test query to any session that is not exposed. The test session returns a real or randomly generated key each with probability half and the adversary must guess whether the returned key is real. The adversary is called successful if it can distinguish the real key with probability significantly larger than 1/2.

An instance of protocol execution results in a partnership or agreement between two oracle instances of principals' communication. Partnership is a tricky notion, because the identity of the parties can be unknown before the protocol instance completes. In [18] Canetti and Krawczyk accommodate this by allowing the communicating oracles to learn the identity of the other party from the protocol instance.

Definition 1. *Suppose oracle $\mathcal{O}_{s,P}$ has accepted with output (P, s, Q). The oracle $\mathcal{O}_{s,R}$ is called the* matching session *of $\mathcal{O}_{s,P}$ if either*

1. *$\mathcal{O}_{s,R}$ has accepted with output (R, s, S), where $R = Q$ and $S = P$, or,*
2. *$\mathcal{O}_{s,R}$ has not completed.*

With this background, it is possible to define security in the model.

Definition 2. *A protocol π is called* SK-secure *if for all SK-adversaries $\mathcal{A}$ the following holds:*

1. *If two uncorrupted parties P and Q complete matching sessions $\mathcal{O}_{s,P}$ and $\mathcal{O}_{s,Q}$ of π with public outputs (P, s, Q) and (Q, s, P), respectively, then the session key K output in these sessions is the same except with a negligible probability.*
2. *$\mathcal{A}$ succeeds in distinguishing the output from its test query with probability not more than 1/2 plus a negligible fraction.*

3.2 Security Model for Key Exchange with Anonymous Authentication

In this paper, we first consider unilateral anonymous authentication and use DAA as the fundamental building block for anonymous authentication. The case of mutual anonymous authentication and use of group signatures shall be discussed in Section 6. In our model, we always assume that the protocol initiator P has a DAA private key and the protocol responder Q has a regular signature key. In the example in Section 1 where a trusted platform mutual authenticates with an Internet server, it is natural to have the anonymous entity as the protocol initiator. If the adversary starts a session with the protocol initiator who does not have a DAA private key or with the protocol responder who does not have a regular signature key, the session would fail.

Note that anonymous authentication in our model is different from anonymity without authentication, where P does not show any identity or signatures to Q. Anonymous authentication in our context means that P authenticates to Q as a member of a DAA group created by an issuer I, but Q does not learn the actual identity of P.

Canetti and Krawczyk prove that the SIGMA protocol in Section 2.3 is secure in the key exchange model described above. However, this security model assumes concrete identity of each party revealed during the protocol, so we will need to extend their model to accommodate anonymous authentication where the actual identity of the protocol initiator cannot be disclosed. The technical need is to have a specific identity to use to establish the matching sessions property. In particular, each party P must verify that the peer Q is the same entity throughout an instance of the protocol. To this end we make the following

Definition 3. *Suppose P is a member of a group identified by ID_I, and let $f(x)$ denote a one-way function. A pseudo identity of P is a pair $(\text{ID}_I, f(n))$, where n is a randomly selected nonce.*

We substitute anonymous identity instances in place of concrete identities in the SIGMA protocol. A principal P outputs $(\hat{P}, s, Q)$ where $\hat{P} = (\text{ID}_I, f(n))$, and argue that this is an appropriate identity for the Canetti-Krawczyk key exchange model. Indeed, P's DAA signature proves its membership in a specific DAA group, but this by itself does not prove that the messages of the protocol instance are exchanged with one single group member. To accomplish this, the definition of pseudo identity requires P to send a commitment $f(n)$ of the random value n to its peer Q, and then P proves it knows n as part of authentication. The allows the matching sessions property to be established, because an adversary cannot arrange for a second group member R to interject itself into the protocol, because R does not know the value of n. We now define security of key exchange with anonymous authentication in below.

Definition 4. *A key exchange protocol with anonymous authentication π is secure if it is SK-secure and anonymous.*

The property of SK-security follows the Canetti-Krawczyk key exchange model and is defined as follows.

Definition 5. *A key exchange protocol with anonymous authentication π is* SK-secure *if for all polynomial-time SK-adversaries $\mathcal{A}$ the following holds:*

1. *If two uncorrupted parties P and Q complete matching sessions $\mathcal{O}_{s,P}$ and $\mathcal{O}_{s,Q}$ of π with public outputs $(\hat{P}, s, Q)$ and $(Q, s, \hat{P})$, respectively, where $\hat{P}$ is the pseudo identity of P, then the session key K for these two sessions is the same except with negligible probability.*
2. *$\mathcal{A}$ succeeds in distinguishing the output from its test query with probability no more than 1/2 plus a negligible fraction.*

We now define the anonymous property of a key exchange protocol with anonymous authentication. Roughly speaking, a key exchange protocol π is anonymous if the protocol initiator P remains anonymous to its peer in the key exchange session. The definition of anonymity follows the definition of the underlying anonymous digital signature schemes used in the protocol. For example using DAA, we define the anonymity of a key exchange protocol π by following the user-controlled-anonymity property of DAA in Appendix A. General speaking, a key exchange protocol with anonymous authentication π is anonymous if there is no probabilistic polynomial-time adversary can win the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows:

– Initial: $\mathcal{C}$ assigns each entity a unique DAA private key. $\mathcal{C}$ also assigns a unique identity and a regular signature key to $\mathcal{A}$.
– Phase 1: $\mathcal{C}$ is probed by $\mathcal{A}$ who makes the following queries:
  - Key exchange. $\mathcal{A}$ submits an entity's identity P of his choice to $\mathcal{C}$, who runs a key exchange protocol π using P as the protocol initiator with $\mathcal{A}$ as the protocol responder.
  - Corrupt. $\mathcal{A}$ submits an entity's identity P of his choice to $\mathcal{C}$, who responds with the DAA private key of P.
– Challenge: At the end of Phase 1, $\mathcal{A}$ chooses two entities P_0 and P_1 of his choice. $\mathcal{A}$ must not have made any corrupt query on either P_0 or P_1. To make the challenge, $\mathcal{C}$ chooses a bit b uniformly at random, then runs a key exchange protocol π using P_b as the protocol initiator with $\mathcal{A}$ as the protocol responder.
– Phase 2: $\mathcal{A}$ continues to probe $\mathcal{C}$ with the same type of queries that it made in Phase 1. However, he is not allowed to corrupt either P_0 or P_1.
– Response: $\mathcal{A}$ returns a bit b'. We say that the adversary wins the game if $b = b'$.

Definition 6. *Let $\mathcal{A}$ denote an adversary that plays the game above. A key exchange protocol π is* anonymous *if for any polynomial-time adversary $\mathcal{A}$, the probability of $\mathcal{A}$ in breaking the anonymity game is no more than 1/2 plus a negligible fraction.*

Note that the above definition of anonymity is generic to any group-based anonymous digital signature schemes. We can modify the above definition to be DAA specific such that the identity of the protocol initiator is unrevealed in the protocol but could be linkable under the entity's control. Similarly, we can modify the above definition for group signatures such that the identity of the protocol initiator is anonymous but openable by a trusted group manager.

4 Proposed DAA-SIGMA Protocol

We now describe our DAA-SIGMA key exchange protocol as follows. Our key exchange protocol builds on top of the SIGMA protocol with the following changes: (1) P uses its DAA private key to signs the ephemeral DH public keys and (2) P uses $(\textsc{id}_I, g^x)$ as its pseudo identity in the protocol where the one-way function is defined as $f(x) = g^x$.

We assume the following initial information. Let P and Q be the two parties of the DAA-SIGMA key exchange protocol, where P is the protocol initiator who activates the session and Q is the intended peer of the session.

- P has a DAA signing key issued by an issuer I whose identity is $\textsc{id}_I$. P can use the DAA signing algorithm DAA-SIG to create DAA signatures.
- Q has an identity $\textsc{id}_Q$ and a public-private key pair. Q can use his private key and a signature algorithm SIG to create a signature.
- Both entities agree on G, a cyclic group of prime order q in which the DDH problem is hard, and g, a generator of G. For example, one can choose primes p and q such that $q|p-1$, and then choose a number g whose multiplicative order modulo p is q.
- The protocol uses a message authentication code family MAC and a pseudorandom function family PRF.

The DAA-SIGMA protocol has the following messages and steps.

$$\begin{aligned}
&\text{Message 1 } (P \rightarrow Q): \ s, g^x \\
&\text{Message 2 } (P \leftarrow Q): \ s, g^y, \textsc{id}_Q, \textsc{mac}_{k_1}(s, \textsc{id}_Q), \textsc{sig}_Q(s, g^x, g^y) \\
&\text{Message 3 } (P \rightarrow Q): \ s, \textsc{id}_I, g^x, \textsc{mac}_{k_1}(s, \textsc{id}_I, g^x), \textsc{daa-sig}_P(s, g^y, g^x)
\end{aligned}$$

Sending Message 1

1. P chooses a session id s. Alternatively, session id s can be pre-negotiated by P and Q.
2. P picks a random $x \leftarrow \mathbb{Z}_q$ and computes its ephemeral DH public key g^x.
3. P sends $\{s, g^x\}$ to Q.

Sending Message 2

1. Q picks a random $y \leftarrow \mathbb{Z}_q$ and computes its ephemeral DH public key g^y.
2. Q computes $g^{xy} = (g^x)^y$.
3. Q derives two keys $k_0 = \textsc{prf}_{g^{xy}}(0)$, and $k_1 = \textsc{prf}_{g^{xy}}(1)$.
4. Q securely erases y and g^{xy} from its memory.
5. Q computes $\textsc{mac}_{k_1}(s, \textsc{id}_Q)$.
6. Q computes $\textsc{sig}_Q(s, g^x, g^y)$ using its private key.
7. Q sends $\{s, g^y, \textsc{id}_Q, \textsc{mac}_{k_1}(s, \textsc{id}_Q), \textsc{sig}_Q(s, g^x, g^y)\}$ to P.

Receiving Message 2

1. P computes $g^{xy} = (g^y)^x$.
2. P derives two keys $k_0 = \textsc{prf}_{g^{xy}}(0)$, and $k_1 = \textsc{prf}_{g^{xy}}(1)$.
3. P securely erases x and g^{xy} from its memory.
4. P verifies $\textsc{mac}_{k_1}(s, \textsc{id}_Q)$ using derived key k_1.
5. P retrieves the public key of Q from $\textsc{id}_Q$.
6. P verifies $\textsc{sig}_Q(s, g^x, g^y)$ using Q's public key.
7. If one of the above verification steps fails, the session is aborted and outputs "failure".

Sending Message 3

1. P computes $\textsc{mac}_{k_1}(s, \textsc{id}_I, g^x)$.
2. P uses its DAA signing key to compute $\textsc{daa-sig}_P(s, g^y, g^x)$.
3. P sends $\{s, \textsc{id}_I, g^x, \textsc{mac}_{k_1}(s, \textsc{id}_I, g^x), \textsc{daa-sig}_P(s, g^y, g^x)\}$ to Q.
4. P completes the session with public output $((\textsc{id}_I, g^x), s, \textsc{id}_Q)$ and secret session key k_0.

Receiving Message 3

1. Q verifies $\textsc{mac}_{k_1}(s, \textsc{id}_I, g^x)$ using derived key k_1.
2. Q retrieves the DAA group public key of the issuer I from $\textsc{id}_I$.
3. Q verifies $\textsc{daa-sig}_P(s, g^y, g^x)$ using the DAA group public key.
4. If one of the above verification steps fails, the session is aborted and outputs of "failure".
5. Q completes the session with public output $(\textsc{id}_Q, s, (\textsc{id}_I, g^x))$ and secret session key k_0.

5 Security Proof of the DAA-SIGMA Protocol

In Theorem 6 of [19] Canetti and Krawczyk prove that the SIGMA protocol is secure in their model. Our security proof of the DAA-SIGMA protocol follows their security proof. We replace the signing algorithm of the protocol initiator P with the DAA signing algorithm and P's identity with the pseudo identity $(\textsc{id}_I, g^x)$, where $\textsc{id}_I$ is the identity of the issuer and g^x is an ephemeral Diffie-Hellman value P generates. To prove the security of our DAA-SIGMA protocol, we begin with a review of the definition of the Decisional Diffie-Hellman (DDH) assumption, which is the assumption underlying the security of all Diffie-Hellman based key exchange protocols.

Assumption 1 *Let G, generated by g, be a cyclic group of prime order q. The DDH assumption in G holds if the probability distributions of $Q_0 = \{\langle g, g^x, g^y, g^{xy}\rangle : x, y \leftarrow \mathbb{Z}_q\}$ and $Q_1 = \{\langle g, g^x, g^y, g^r\rangle : x, y, r \leftarrow \mathbb{Z}_q\}$ are computationally indistinguishable.*

In addition to the DDH assumption, the security of the key exchange protocol in Section 4 also depends on the security of the underlying cryptographic primitives, namely, digital signatures, message authentication codes, pseudo-random functions, and DAA. We have the following theorem.

Theorem 1. *Under the DDH assumption in G and assuming the security of the underlying cryptographic functions* SIG, MAC, PRF, DAA-SIG, *the* DAA-SIGMA *protocol in Section 4 is* secure *in the model defined in Section 3.*

To prove the above theorem, we prove the following two lemmas.

Lemma 1. *If the underlying DAA scheme is secure, the* DAA-SIGMA *protocol is* anonymous *in the model defined in Section 3.2.*

Proof. In the DAA-SIGMA protocol, the protocol initiator P reveals s and g^x to its peer in the first message, and reveals s, $\textsc{id}_I$, g^x, $\textsc{mac}_{k_1}(s, \textsc{id}_I, g^x)$, and $\textsc{daa-sig}_P(s, g^y, g^x)$ in the final message. Since s and g^x are randomly chosen at each session, if the DAA signature is anonymous, P only reveals its pseudo identity $(\textsc{id}_I, g^x)$ to its peer, but not the real identity. To prove the DAA-SIGMA protocol is anonymous, we use the following reduction. Suppose there exists a polynomial-time adversary $\mathcal{A}$ that breaks the anonymity game in Section 3.2, we can construct an algorithm $\mathcal{B}$ that breaks the user-controlled-anonymity of DAA defined in Appendix A by interacting with $\mathcal{A}$ as follows:

- $\mathcal{B}$ received `isk` and `gpk` from the challenger.

– $\mathcal{B}$ creates n entities, but does not assign any DAA private key to each entity.
– $\mathcal{B}$ chooses an identity ID_I for the DAA group and sends ID_I and $\mathtt{gpk}$ to $\mathcal{A}$.
– $\mathcal{B}$ picks a unique identity and a regular signature key for $\mathcal{A}$.
– $\mathcal{B}$ is probed by $\mathcal{A}$ with following two types of queries:
  - $\mathcal{A}$ makes a key exchange query on an entity P. If P has not been queried before, $\mathcal{B}$ first runs a join query with the challenger to create a DAA private key for P. Note that $\mathcal{B}$ does not know the DAA private key of P. $\mathcal{B}$ simulates the DAA-SIGMA protocol with $\mathcal{A}$ as follows:
    1. $\mathcal{B}$ randomly chooses a session id s and x. $\mathcal{B}$ computes g^x and sends s and g^x to $\mathcal{A}$.
    2. $\mathcal{B}$ receives the DAA-SIGMA message 2 from $\mathcal{A}$ and validates the message.
    3. $\mathcal{B}$ computes g^{xy} and derives k_0 and k_1 by following the DAA-SIGMA protocol.
    4. $\mathcal{B}$ computes $\text{MAC}_{k_1}(s, \text{ID}_I, g^x)$. $\mathcal{B}$ then makes a sign query on P to the challenger with the message $m = (s, g^y, g^x)$ and obtains $\text{DAA-SIG}_P(s, g^y, g^x)$.
    5. $\mathcal{B}$ sends s, ID_I, g^x, $\text{MAC}_{k_1}(s, \text{ID}_I, g^x)$, and $\text{DAA-SIG}_P(s, g^y, g^x)$ to $\mathcal{A}$.
  - $\mathcal{A}$ makes a corrupt query on an entity P. If P has not been queried before, $\mathcal{B}$ first runs a join query with the challenger to create a DAA private key for P. $\mathcal{B}$ then makes a corrupt query to the challenger to obtain P's DAA private key. $\mathcal{B}$ forwards the private key to $\mathcal{A}$.
– $\mathcal{B}$ receives the challenge from $\mathcal{A}$ with entities P_0 and P_1. $\mathcal{B}$ makes sure both entities have not been corrupted. $\mathcal{B}$ simulates the DAA-SIGMA protocol with $\mathcal{A}$ as in the key exchange query above, except in step 4, $\mathcal{B}$ makes a challenge query to the challenger with P_0, P_1, and $m = (s, g^y, g^x)$ as input and obtains a DAA signature σ on (s, g^y, g^x) using P_b's DAA private key, where b is unknown to $\mathcal{B}$. In the end, $\mathcal{B}$ sends s, ID_I, g^x, $\text{MAC}_{k_1}(s, \text{ID}_I, g^x)$, and σ to $\mathcal{A}$ to simulate the message 3 of the DAA-SIGMA protocol.
– $\mathcal{A}$ can continue make key exchange and corrupt queries to $\mathcal{B}$ as far as $\mathcal{A}$ does not make corrupt queries on either P_0 or P_1.
– $\mathcal{A}$ outputs b' by guessing b. $\mathcal{B}$ outputs the same b'.

Let ϵ be the probability that $\mathcal{A}$ succeeds in breaking the anonymity game of the key exchange protocol. $\mathcal{B}$ has the same probability ϵ in breaking the user-controlled-anonymity game of DAA.

Lemma 2. *Under the DDH assumption in G and assuming the security of the underlying cryptographic functions* SIG, MAC, PRF, DAA-SIG, *the* DAA-SIGMA *protocol in Section 4 is SK-secure in the model defined in Section 3.2.*

The proof of this lemma follows the security proof of the SIGMA protocol in [19]. Let us use the term Σ-attacker to denote an SK-adversary attacking against the DAA-SIGMA protocol. We need to prove the two properties of secure key exchange protocol defined in Definition 5.

P1. If two uncorrupted parties P and Q complete matching sessions $((\text{ID}_I, g^x), s, \text{ID}_Q)$ and $(\text{ID}_Q, s, (\text{ID}_I, g^x))$ respectively under the DAA-SIGMA protocol then, except for a negligible probability, the session key output in these sessions is the same.

P2. No efficient Σ-attacker can distinguish a real response from a random response with non-negligible probability. More precisely, if a given Σ-attacker we define
1. $P_{\text{REAL}}(\mathcal{A}) = \Pr[\mathcal{A}$ outputs 1 when given the real test session key$]$
2. $P_{\text{RANDOM}}(\mathcal{A}) = \Pr[\mathcal{A}$ ouptus 1 when given a random test session key$]$

then we need to prove that for any Σ-attacker $\mathcal{A}$ that $|P_{\text{real}}(\mathcal{A}) - P_{\text{random}}(\mathcal{A})|$ is negligible.

For the proof we assume that the Σ-attacker $\mathcal{A}$ never exposes the test session, which would represent an attack that does not undermine the protocol directly.

5.1 Proof of Property P1

The proof for the first property in [19] goes over unchanged if we substitute our notion of a pseudo identity for the identity of the initiator and if we replace the initiator's signature scheme with DAA. Let $\mathcal{A}$ be a Σ-attacker and let P and Q be two uncorrupted entities that complete matching sessions $((\text{ID}_I, g^x), s, \text{ID}_Q)$ and $(\text{ID}_Q, s, (\text{ID}_I, g^x))$. We want to prove that both sessions output the same session key k_0 regardless of $\mathcal{A}$'s operations. It is enough to show that both entities compute the same DH value g^{xy} from which k_0 is derived.

Let us use u_P to denote the DH public key send in the message 1 by P where $u_P = g^x$ with x chosen by P, and v_P to denote the DH public key received in the message 2 of session s. Similarly, let u_Q be the DH public key received by Q in the message 1 and v_Q be the DH public key send by Q in the message 2 where $v_Q = g^y$ with y chosen by Q. The signature produced by Q during the session s is $\text{SIG}_Q(s, u_Q, v_Q)$, while the signature that P verifies the message 2 is $\text{SIG}_Q(s, u_P, v_P)$. As s is unique per session, the first signature is the only one that Q ever produces with the value s as session id, then it must be that either all arguments of the first and second signatures are the same, or $\mathcal{A}$ creates a forgery on the second signature. As we assume SIG is a secure signature scheme, it follows that except for a negligible probably, $u_P = u_Q$ and $v_P = v_Q$. Now the DH value computed by P is $v_P^x = v_Q^x = (g^y)^x = g^{xy}$, while the DH value computed by Q is $u_Q^y = u_P^y = (g^x)^y = g^{xy}$. Thus both compute the same session key. This concludes our proof of property P1.

5.2 Proof of Property P2

The strategy in [19] to prove the second property is to show that any attacker who can distinguish the real key from a random key with non-negligible probability can be used to construct a second attacker against the DDH assumption or one of the underlying primitives – the signature scheme SIG, the MAC scheme MAC, or the pseudo-random function PRF. In our setting, we must replace the initiator's signature with a DAA signature, and its identity with a pseudo identity.

We now sketch the proof as follows. We show that any Σ-attacker $\mathcal{A}$ that succeeds in distinguishing between a real and a random response to the test session query, we can build a DDH distinguisher $\mathcal{D}$ that distinguishes triple (g^x, g^y, g^{xy}) from random triples (g^x, g^y, g^r) with the same success advantage as $\mathcal{A}$, or there is an algorithm that we can construct explicitly that breaks one of the underlying cryptographic primitives (i.e., SIG, MAC, PRF, or DAA-SIG). The distinguisher $\mathcal{D}$ gets input (g^x, g^y, z) where z is either g^{xy} or g^r for $r \leftarrow \mathbb{Z}_q$. $\mathcal{D}$ starts by simulating a run of $\mathcal{A}$ on a virtual instantiation of the DAA-SIGMA protocol Σ and use the values g^x and g^y from the input triple as the DH public keys in the messages 1 and 2 of a randomly chosen session denoted s_0, initiated by $\mathcal{A}$ in this execution of the DAA-SIGMA protocol. If $\mathcal{A}$ happens to choose this session s_0 as its test session, then $\mathcal{D}$ can provide $\mathcal{A}$ with z as the response to the test query. In this case, if $\mathcal{A}$ outputs that the response was real then $\mathcal{D}$ will decide that $z = g^{xy}$, otherwise, $\mathcal{D}$ will decide that z is random.

One difficulty with the above strategy is that $\mathcal{D}$ actually changes the regular behavior of the parties in session s_0, e.g., it uses the value z to derive the key k_1 used in the MAC function, we have to show that the original ability of $\mathcal{A}$ to distinguish between "real" and "random"is not significantly reduced by the simulation changes. Canetti and Krawczyk [19] addressed the difficulty by defining a sequence of several simulators as follows which differ from each other by the way they choose k_0

and k_1 used in the simulation of the s_0 session.

$$\begin{aligned}
\hat{\mathcal{S}}\text{-REAL}: &\quad k_0 \leftarrow \text{PRF}_{g^{xy}}(0),\ \ k_1 \leftarrow \text{PRF}_{g^{xy}}(1) \\
\hat{\mathcal{S}}\text{-RPRF}: &\quad k_0 \leftarrow \text{PRF}_{k}(0),\ \ k_1 \leftarrow \text{PRF}_{k}(1),\ \ k \leftarrow \mathit{random}() \\
\hat{\mathcal{S}}\text{-ALLR}: &\quad k_0 \leftarrow \mathit{random}(),\ \ k_1 \leftarrow \mathit{random}() \\
\hat{\mathcal{S}}\text{-HYBR}: &\quad k_0 \leftarrow \mathit{random}(),\ \ k_1 \leftarrow \text{PRF}_{k}(1),\ \ k \leftarrow \mathit{random}() \\
\hat{\mathcal{S}}\text{-RAND}: &\quad k_0 \leftarrow \mathit{random}(),\ \ k_1 \leftarrow \text{PRF}_{g^{xy}}(1)
\end{aligned}$$

$\hat{\mathcal{S}}$-REAL is the simulator that corresponds to a "real" run of $\mathcal{A}$ while $\hat{\mathcal{S}}$-RAND corresponds to a "random" experiment where the session key in session s_0 provided to $\mathcal{A}$ is chosen as a random and independent value k_0. The rest of the simulators are "hybrid" simulators. It was shown in [19] that either all the distributions generated by these simulators are computationally indistinguishable, or that a successful distinguisher against DDH or against PRF family exists. From this we get a proof that $\hat{\mathcal{S}}$-REAL and $\hat{\mathcal{S}}$-RAND are actually indistinguishable, and we conclude that the values P_{RANDOM} and P_{REAL} differ by at most a negligible quantity.

In the security proof of the second property by [19], several properties of the protocol are shown that related to the authentication elements such as signatures (Lemma 7 in [19]) and MAC (Lemma 11 in [19]). In the DAA-SIGMA protocol, we replace the protocol initiator's signature with a DAA signature. We can show that

– If $\mathcal{A}$ does not have access to any legitimate DAA private keys, then Lemma 7.4 in [19] can be used to break the unforgeability property of DAA.
– If $\mathcal{A}$ does control some valid DAA private keys, then the commitment in the pseudo identity guarantees that the adversary cannot break the session key of two uncorrupted parties unless he can break DDH assumption.

The above arguments easily follow the proof in [19]. The details will be given in the full paper.

6 Discussions and Extensions

In this section, we discuss several variants and possible extensions of the our DAA-SIGMA protocol. We begin with how to use group signature in the SIGMA key exchange protocol.

6.1 Using Group Signatures Instead of DAA

Although this paper primarily focuses on DAA, our DAA-SIGMA protocol can be easily extended to support group signatures [2, 5, 6, 29]. Both DAA and group signatures are group-based anonymous signature schemes. That is, there is an issuer in both schemes who creates a group of members and a group public key. Each group member has an individual private key. A DAA signature or a group signature can be verified using the corresponding group public key. Group signatures are different from DAA in that group signatures can be opened by a trusted authority and the identity of the actual signer can be extracted, whereas DAA signature cannot be opened.

To use a group signature scheme in the key exchange protocol for anonymous authentication, we can replace the DAA signature in the DAA-SIGMA protocol with a group signature. We called the modification as GS-SIGMA protocol. More specifically, we assume party P has a group private key issued by an issuer I. We use $\textsc{id}_I$ to denote the identity of the issuer and $\textsc{gs-sig}_P(m)$ to denote a group signature on m created by P. The GS-SIGMA protocol has the following messages

$$\begin{aligned}
\text{Message 1 } (P \rightarrow Q):&\ s, g^x \\
\text{Message 2 } (P \leftarrow Q):&\ s, g^y, \textsc{id}_Q, \textsc{mac}_{k_1}(s, \textsc{id}_Q), \textsc{sig}_Q(s, g^x, g^y) \\
\text{Message 3 } (P \rightarrow Q):&\ s, \textsc{id}_I, g^x, \textsc{mac}_{k_1}(s, \textsc{id}_I, g^x), \textsc{gs-sig}_P(s, g^y, g^x)
\end{aligned}$$

This GS-SIGMA protocol is secure in the key exchange model for anonymous authentication defined in Section 3, if the DDH assumption holds and if the underlying SIG, MAC, PRF, and GS-SIG functions are secure. The security proof is straight-forward from Section 5.

6.2 Use Certificates Instead of Identities

In the DAA-SIGMA protocol presented in Section 4, $\textsc{id}_I$ and $\textsc{id}_Q$ represent the real identities of the issuer and the party Q, respectively. We assume that P can obtain Q's public key using $\textsc{id}_Q$, and similarly Q can obtain P's DAA public key from the identity of the issuer $\textsc{id}_I$. In practice, we can replace the identities with full certificates signed by a trusted Certificate Authority (CA). More specifically, assume all the parties share the public key of the trusted CA. If an entity has a regular public and private key pair, we assume it has a public-key certificate issued by the trusted CA. If an entity has a DAA private key, we assume it has a certificate issued by the trusted CA to certify the issuer and the DAA public key, denoted as $\textsc{cert}_I$. The certificate-based DAA-SIGMA protocol is presented as follows.

$$\begin{aligned}
\text{Message 1 } (P \rightarrow Q):&\ s, g^x \\
\text{Message 2 } (P \leftarrow Q):&\ s, g^y, \textsc{cert}_Q, \textsc{mac}_{k_1}(s, \textsc{cert}_Q), \textsc{sig}_Q(s, g^x, g^y) \\
\text{Message 3 } (P \rightarrow Q):&\ s, \textsc{cert}_I, g^x, \textsc{mac}_{k_1}(s, \textsc{cert}_I, g^x), \textsc{daa-sig}_P(s, g^y, g^x)
\end{aligned}$$

Note that, in the above protocol, the protocol participants P and Q do not need to retrieve public keys from the identities. Instead, they can use the public keys embedded in the certificate to verify the signatures in messages 2 and 3. Canetti and Krawczyk proved the certificate-based SIGMA protocol is secure in [19]. The security proofs for our certificate-based DAA-SIGMA protocol should work as well.

6.3 Supporting User-Controlled-Traceability and Revocation of DAA

As we mentioned earlier, the DAA signing algorithm takes the verifier's basename `bsn` as an optional input. If the basename is not given in the signing algorithm, we treat $\texttt{bsn} = \perp$. DAA signatures are unlinkable if $\texttt{bsn} = \perp$ and linkable otherwise. This feature is called user-controlled-traceability in [9], as the signer and verifier can negotiate whether to use basename. The DAA-SIGMA protocol in Section 4 only supports unlinkable DAA signatures, but it can be easily extended to support user-controlled-traceability by passing the basename value in the protocol messages.

Another important feature of DAA is revocation. A revocation method in DAA proposed in [10, 12] is called signature-based revocation, where the issuer can revoke a DAA signer without knowing the signer's private key. In this revocation method, the issuer puts DAA signatures that are suspicious or were involved in malicious transactions to a revocation list denoted as `sRL`. The verifier can request the signer to prove that his DAA signing key did not generate those signatures in the revocation list. To support this type of revocation, the verifier needs to sends the revocation list to the signer so that signer can prove innocent. To support this revocation method in the DAA-SIGMA protocol, the revocation list $\texttt{sRL}$ needs to be passed in the protocol messages.

We now extend the DAA-SIGMA protocol to support the above two functionalities of DAA as follows. In the following protocol, variables in brackets are optional fields.

$$\begin{aligned}
&\text{Message 1 } (P \to Q): \ s, g^x, [\textsc{id}_I] \\
&\text{Message 2 } (P \leftarrow Q): \ s, g^y, \textsc{id}_Q, [\texttt{bsn}_Q, \texttt{sRL}], \textsc{mac}_{k_1}(s, \textsc{id}_Q, [\texttt{bsn}_Q, \texttt{sRL}]), \textsc{sig}_Q(s, g^x, g^y) \\
&\text{Message 3 } (P \to Q): \ s, \textsc{id}_I, g^x, \textsc{mac}_{k_1}(s, \textsc{id}_I, g^x), \textsc{daa-sig}_P(s, g^y, g^x)
\end{aligned}$$

Note that without the optional fields (i.e., $\textsc{id}_I$ in the message 1 and $\texttt{bsn}_Q$ and $\texttt{sRL}$ in the message 2), the above protocol is exact the same DAA-SIGMA protocol as in Section 4. To support user-control-traceability feature of DAA, Q can send its basename $\texttt{bsn}_Q$ in the message 2 so that P can create the DAA signature with $\texttt{bsn}_Q$ as input. To support signature-based revocation, P first reveals its DAA group (in this case, the identity of the issuer $\textsc{id}_I$) to Q who can find the corresponding revocation list $\texttt{sRL}$ of the DAA group. Q then sends $\texttt{sRL}$ to P in the message 2 so that P can prove he has not been revoked in the message 3 as part of the DAA signature.

6.4 Mutual Anonymous Authentication using DAA

The DAA-SIGMA protocol presented in Section 4 is a key exchange protocol with unilateral anonymous authentication. Take the example in Section 1, it is natural to have the protocol initiator as the anonymous entity. It may not make sense for mutual anonymous authentication over the Internet, as the protocol initiator needs to know who is the protocol responder in order to know where to send the first message. Mutual anonymous authentication may be useful for ubiquitous computing environments or for vehicle communication where two entities (e.g., mobile devices or cars) are already physically close to each other and want to authenticate to each other anonymously.

We can extend the DAA-SIGMA protocol to support mutual anonymous authentication as follows. Assume each protocol party have a DAA private key for computing anonymous signatures, i.e., P has a DAA key issued by an issuer I and Q has a DAA key issued by an issue I', where I and I' could be the same entity or two separate entities. The messages 1 and 3 are the same as in the Section 4. In message 2, Q uses $(\textsc{id}_{I'}, g^y)$ as its pseudo identity instead of revealing its actual identity $\textsc{id}_Q$ in order to be anonymous in the key exchange.

$$\begin{aligned}
&\text{Message 1 } (P \to Q): \ s, g^x \\
&\text{Message 2 } (P \leftarrow Q): \ s, g^y, \textsc{id}_{I'}, \textsc{mac}_{k_1}(s, \textsc{id}_{I'}, g^y), \textsc{daa-sig}_Q(s, g^x, g^y) \\
&\text{Message 3 } (P \to Q): \ s, \textsc{id}_I, g^x, \textsc{mac}_{k_1}(s, \textsc{id}_I, g^x), \textsc{daa-sig}_P(s, g^y, g^x)
\end{aligned}$$

7 Conclusions and Future Work

We presented in this paper a new security model for key exchange with anonymous authentication derived from the Canetti-Krawczyk key exchange model. We proposed a modification to the SIGMA key exchange protocol with DAA incorporated and with the notion of pseudo identity to achieve anonymous authentication. We proved our DAA-SIGMA key exchange protocol is secure under the new security model. Future related research topics include

1. Provide a formal security model for mutual anonymous authentication and prove our mutual anonymous authentication protocol in Section 6 is secure.
2. Study how to incorporate DAA into SSL/TLS protocol with provable security under our new security model.

Acknowledgement

We thank Liqun Chen for her helpful discussions and feedback.

References

1. ISO/IEC PAS DIS 11889: Information technology – Security techniques – Trusted platform module.
2. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — CRYPTO '00*, volume 1880 of *LNCS*, pages 255–270. Springer, 2000.
3. Shane Balfe, Amit D. Lakhani, and Kenneth G. Paterson. Trusted computing: Providing security for peer-to-peer networks. In *Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing*, pages 117–124, August 2005.
4. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology — CRYPTO '93*, volume 773 of *LNCS*, pages 232–249. Springer, 1993.
5. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology — CRYPTO '04*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.
6. Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of 11th ACM Conference on Computer and Communications Security*, pages 168–177, October 2004.
7. Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 132–145. ACM Press, 2004.
8. Ernie Brickell, Liqun Chen, and Jiangtao Li. A new direct anonymous attestation scheme from bilinear maps. In *Proceedings of 1st International Conference on Trusted Computing*, volume 4968 of *LNCS*, pages 166–178. Springer, 2008.
9. Ernie Brickell, Liqun Chen, and Jiangtao Li. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. *International Journal of Information Security*, 8(5):315–330, 2009.
10. Ernie Brickell and Jiangtao Li. Enhanced Privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 6th ACM Workshop on Privacy in the Electronic Society*, pages 21–30, October 2007.
11. Ernie Brickell and Jiangtao Li. Enhanced Privacy ID: A remote anonymous attestation scheme for hardware devices. *Intel Technology Journal: Advances in Internet Security*, 13(2), 2009.
12. Ernie Brickell and Jiangtao Li. Enhanced Privacy ID from bilinear pairing for hardware authentication and attestation. In *Proceedings of 2nd IEEE International Conference on Information Privacy, Security, Risk and Trust*, 2010.
13. Ernie Brickell and Jiangtao Li. Enhanced Privacy ID: Hardware attestation beyond TPM. First Workshop on Anonymous Digital Signatures: Mechanisms & Usages, 2010. `http://www.trust2010.org/workshop-anon.html`.
14. Ernie Brickell and Jiangtao Li. A pairing-based DAA scheme further reducing TPM resources. In *Proceedings of 3rd International Conference on Trust and Trustworthy Computing*, volume 6101 of *LNCS*, pages 181–195. Springer, 2010.
15. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology — EUROCRYPT '01*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.
16. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Proceedings of the 3rd Conference on Security in Communication Networks*, volume 2576 of *LNCS*, pages 268–289. Springer, 2002.
17. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — CRYPTO '04*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.
18. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology — EUROCRYPT '01*, volume 2045 of *LNCS*, pages 453–474. Springer, 2001.

19. Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *Advances in Cryptology — CRYPTO '02*, volume 2442 of *LNCS*, pages 143–161. Springer, 2002.
20. Emanuele Cesena, Hans Löhr, Gianluca Ramunno, Ahmad-Reza Sadeghi, and Davide Vernizzi. Anonymous authentication with tls and daa. In *Proceedings of 3rd International Conference on Trusted and Trustworthy Computing*, volume 6101 of *LNCS*, pages 47–62. Springer, 2010.
21. Zhenchuan Chai, Zhenfu Cao, and Rongxing Lu. Efficient password-based authentication and key exchange scheme preserving user privacy. In *1st International Conference on Wireless Algorithms, Systems, and Applications*, volume 4138 of *LNCS*, pages 467–477, 2006.
22. David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.
23. Liqun Chen, Paul Morrissey, and Nigel P. Smart. Pairings in trusted computing. In *Proceedings of the 2nd Internation Conference on Pairing-Based Cryptography*, volume 5209 of *LNCS*, pages 1–17. Springer, 2008.
24. Liqun Chen, Dan Page, and Nigel P. Smart. On the design and implementation of an efficient DAA scheme. In *Proceedings of the 9th Smart Card Research and Advanced Application IFIP Conference*. Springer, 2010.
25. Xiaofeng Chen and Dengguo Feng. Direct anonymous attestation for next generation TPM. *Journal of Computers*, 3(12):43–50, 2008.
26. Hui Cui and Tianjie Cao. An novel anonymous authenticated and key exchange protocol. *Journal of Networks*, 4(10):985–992, 2009.
27. Anupam Datta, Ante Derek, John C. Mitchell, and Bogdan Warinschi. Computationally sound compositional logic for key exchange protocols. In *Proceedings of 19th IEEE Computer Security Foundations Workshop*, pages 321–334, 2006.
28. Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.
29. Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. *IEICE Transactions*, 89-A(5):1328–1338, 2006.
30. D. Harkins and D. Carrel. The Internet key exchange (IKE). IETF RFC 2409, November 1998.
31. S. Kent and R. Atkinson. Security architecture for the Internet protocol. IETF RFC 2401, November 1998.
32. Adrian Leung and Chris J. Mitchell. Ninja: Non identity based, privacy preserving authentication for ubiquitous environments. In *Proceedings of 9th International Conference on Ubiquitous Computing*, volume 4717 of *LNCS*, pages 73–90. Springer, 2007.
33. Catherine Meadows. Analysis of the Internet key exchange protocol using the NRL protocol analyzer. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 216–231, 1999.
34. Trusted Computing Group. TCG TPM specification 1.2, 2003. `http://www.trustedcomputinggroup.org`.
35. Trusted Computing Group website. `http://www.trustedcomputinggroup.org`.
36. Duong Viet, Akihiro Yamamura, and Hidema Tanaka. Anonymous password-based authenticated key exchange. In *Progress in Cryptology - INDOCRYPT '05*, volume 3797 of *LNCS*, pages 244–257. Springer, 2005.
37. Jing Yang and Zhenfeng Zhang. A new anonymous password-based authenticated key exchange protocol. In *Progress in Cryptology - INDOCRYPT '08*, volume 5365 of *LNCS*, pages 200–212. Springer, 2008.

A Review Security Model of DAA

In this section, we review the specification and security model of DAA proposed in [9]. There are four types of players in a DAA scheme: an issuer $\mathcal{I}$, a TPM $\mathcal{M}_i$, a host $\mathcal{H}_i$ and a verifier $\mathcal{V}_j$. $\mathcal{M}_i$ and $\mathcal{H}_i$ form a platform in the trusted computing environment and share the role of a DAA signer. A DAA scheme has three polynomial-algorithms $(\mathsf{Setup}, \mathsf{Verify}, \mathsf{Link})$ and two interactive protocols $(\mathsf{Join}, \mathsf{Sign})$:

Setup : On input of a security parameter 1^k, $\mathcal{I}$ uses this randomized algorithm to produce a pair $(\mathtt{gpk}, \mathtt{isk})$, where $\mathtt{isk}$ is the issuer's secret key, and $\mathtt{gpk}$ is the public key including the global public parameters.

Join : This randomized algorithm consists of two sub-algorithms $\mathsf{Join_t}$ and $\mathsf{Join_i}$. $\mathcal{M}_i$ uses $\mathsf{Join_t}$ to produce a pair $(\mathtt{sk}_i, \mathtt{comm}_i)$, where $\mathtt{sk}_i$ is the TPM's secret key and $\mathtt{comm}_i$ is a commitment of $\mathtt{sk}_i$. On input of $\mathtt{comm}_i$ and $\mathtt{isk}$, $\mathcal{I}$ uses $\mathsf{Join_i}$ to produce $\mathtt{cre}_i$, which is a DAA credential associated with $\mathtt{sk}_i$. Note that the value $\mathtt{cre}_i$ is given to both $\mathcal{M}_i$ and $\mathcal{H}_i$, but the value $\mathtt{sk}_i$ is known to $\mathcal{M}_i$ only.

Sign : On input of $\mathtt{sk}_i$, $\mathtt{cre}_i$, a basename $\mathtt{bsn}_j$ (the name string of $\mathcal{V}_j$ or a special symbol $\perp$), and a message m that includes the data to be signed and the verifier's nonce n_V for freshness, $\mathcal{M}_i$ and $\mathcal{H}_i$ use this randomized algorithm to produce a signature σ on m under $(\mathtt{sk}_i, \mathtt{cre}_i)$ associated with $\mathtt{bsn}_j$. The basename $\mathtt{bsn}_j$ is used for controlling the linkability.

Verify : On input of m, $\mathtt{bsn}_j$, a candidate signature σ for m, and a set of revoked secret keys $\mathtt{RL}$, $\mathcal{V}_j$ uses this deterministic algorithm to return either 1 (accept) or 0 (reject). How to build the revocation list is out the scope of the DAA scheme.

Link : On input of two signatures σ_0 and σ_1, $\mathcal{V}_j$ uses this deterministic algorithm to return 1 (linked), 0 (unlinked) or $\perp$ (invalid signatures). Link will output $\perp$ if, by using an empty $\mathtt{RL}$, either $\mathsf{Verify}(\sigma_0) = 0$ or $\mathsf{Verify}(\sigma_1) = 0$ holds. Otherwise, Link will output 1 if signatures can be linked or 0 if the signatures cannot be linked.

A DAA scheme is secure if it is correct, user-controlled-anonymous, and user-controlled-traceable.

Correctness If both the signer and verifier are honest, that implies $\mathtt{sk}_i \notin \mathtt{RL}$, the signatures and their links generated by the signer will be accepted by the verifier with overwhelming probability. This means that the DAA scheme must meet the following consistency requirement.

$$\begin{aligned}
&(\mathsf{gpk}, \mathtt{isk}) \leftarrow \mathsf{Setup}(1^k),\ (\mathtt{sk}_i, \mathtt{cre}_i) \leftarrow \mathsf{Join}(\mathtt{isk}, \mathsf{gpk}),\\
&(m_b, \sigma_b) \leftarrow \mathsf{Sign}(m_b, \mathtt{bsn}_j, \mathtt{sk}_i, \mathtt{cre}_i, \mathsf{gpk})|_{b=\{0,1\}},\\
&\Longrightarrow 1 \leftarrow \mathsf{Verify}(m_b, \mathtt{bsn}_j, \sigma_b, \mathsf{gpk}, \mathtt{RL})|_{b=\{0,1\}} \ \wedge\ 1 \leftarrow \mathsf{Link}(\sigma_0, \sigma_1, \mathsf{gpk})|_{\mathtt{bsn}_j \neq \perp}.
\end{aligned}$$

User-Controlled-Anonymity A DAA scheme is user-controlled-anonymous if there is no probabilistic polynomial-time adversary can win the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows:

- Initial: $\mathcal{C}$ runs $\mathsf{Setup}(1^k)$ and gives the resulting $\mathtt{isk}$ and gpk to $\mathcal{A}$.
- Phase 1: $\mathcal{C}$ is probed by $\mathcal{A}$ who makes the following queries:
  - Sign. $\mathcal{A}$ submits a signer's identity S, a basename $\mathtt{bsn}$ (either $\perp$ or a data string) and a message m of his choice to $\mathcal{C}$, who runs Sign to get a signature σ and responds with σ.
  - Join. $\mathcal{A}$ submits a signer's identity S of his choice to $\mathcal{C}$, who runs $\mathsf{Join_t}$ with $\mathcal{A}$ to create $\mathtt{sk}$ and to obtain $\mathtt{cre}$ from $\mathcal{A}$. $\mathcal{C}$ verifies the validation of $\mathtt{cre}$ and keeps $\mathtt{sk}$ secret.
  - Corrupt. $\mathcal{A}$ submits a signer's identity S of his choice to $\mathcal{C}$, who responds with the value $\mathtt{sk}$ of the signer.
- Challenge: At the end of Phase 1, $\mathcal{A}$ chooses two signers' identities S_0 and S_1, a message m and a basename $\mathtt{bsn}$ of his choice to $\mathcal{C}$. $\mathcal{A}$ must not have made any Corrupt query on either S_0 or S_1, and not have made the Sign query with the same $\mathtt{bsn}$ if $\mathtt{bsn} \neq \perp$ with either S_0 or S_1. To make the challenge, $\mathcal{C}$ chooses a bit b uniformly at random, signs m associated with $\mathtt{bsn}$ under $(\mathtt{sk}_b, \mathtt{cre}_b)$ to get a signature σ and returns σ to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ continues to probe $\mathcal{C}$ with the same type of queries that it made in Phase 1. Again, it is not allowed to corrupt any signer with the identity either S_0 or S_1, and not allowed to make any Sign query with $\mathtt{bsn}$ if $\mathtt{bsn} \neq \perp$ with either S_0 or S_1.
- Response: $\mathcal{A}$ returns a bit b'. We say that the adversary wins the game if $b = b'$.

Definition 7. *Let $\mathcal{A}$ denote an adversary that plays the game above. We denote by $\mathbf{Adv}[\mathcal{A}^{anon}_{\mathcal{DAA}}] = |\mathbf{Pr}[b' = b] - 1/2|$ the advantage of $\mathcal{A}$ in breaking the user-controlled-anonymity game. We say that a DAA scheme is user-controlled-anonymous if for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}[\mathcal{A}^{anon}_{\mathcal{DAA}}]$ is negligible.*

User-Controlled-Traceability A DAA scheme is user-controlled-traceable if there is no probabilistic polynomial-time adversary can win the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows:

- Initial: $\mathcal{C}$ executes $\mathsf{Setup}(1^k)$ and gives the resulting $\mathtt{gpk}$ to $\mathcal{A}$. It keeps $\mathtt{isk}$ secret.
- Probing: $\mathcal{C}$ is probed by $\mathcal{A}$ who makes the following queries:
  - Sign. The same as in the game of user-controlled-anonymity.
  - Semi-sign. $\mathcal{A}$ submits a signer's identity S along with the data transmitted from $\mathcal{H}_i$ to $\mathcal{M}_i$ in Sign of his choice to $\mathcal{C}$, who acts as $\mathcal{M}_i$ in Sign and responds with the data transmitted from $\mathcal{M}_i$ to $\mathcal{H}_i$ in the Sign protocol.
  - Join. There are two cases of this query. Case 1: $\mathcal{A}$ submits a signer's identity S of his choice to $\mathcal{C}$, who runs Join to create $\mathtt{sk}$ and $\mathtt{cre}$ for the signer. Case 2: $\mathcal{A}$ submits a signer's identity S with a $\mathtt{sk}$ value of his choice to $\mathcal{C}$, who runs $\mathsf{Join_i}$ to create $\mathtt{cre}$ for the signer and puts the given $\mathtt{sk}$ into $\mathtt{RL}$. $\mathcal{C}$ responds the query with $\mathtt{cre}$. Suppose that $\mathcal{A}$ does not use a single S for both of the cases.
  - Corrupt. This is the same as in the game of user-controlled-anonymity, except that at the end $\mathcal{C}$ puts the revealed $\mathtt{sk}$ into the list of $\mathtt{RL}$.
- Forge: $\mathcal{A}$ returns a signer's identity S, a signature σ, its signed message m and the associated basename $\mathtt{bsn}$. We say that the adversary wins the game if
  1. $\mathsf{Verify}(m, \mathtt{bsn}, \sigma, \mathtt{gpk}, \mathtt{RL}) = 1$ (accepted), but σ is neither a response of the existing Sign queries nor a response of the existing Semi-sign queries (partially); and/or
  2. In the case of $\mathtt{bsn} \neq \perp$, there exists another signature σ' associated with the same identity and $\mathtt{bsn}$, and the output of $\mathsf{Link}(\sigma, \sigma')$ is 0 (unlinked).

Definition 8. *Let $\mathcal{A}$ be an adversary that plays the game above. Let $\mathbf{Adv}[\mathcal{A}^{trace}_{\mathcal{DAA}}] = \mathbf{Pr}[\mathcal{A}$ wins$]$ denote the advantage that $\mathcal{A}$ breaks the user-controlled-traceability game. We say that a DAA scheme is user-controlled-traceable if for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}[\mathcal{A}^{trace}_{\mathcal{DAA}}]$ is negligible.*