ASICS: Authenticated Key Exchange Security Incorporating Certification Systems

Colin Boyd[1], Cas Cremers[2], Michèle Feltz[2], Kenneth G. Paterson[3], Bertram Poettering[3], and Douglas Stebila[1]

[1] Queensland University of Technology, Brisbane, Australia
[2] Institute of Information Security, ETH Zurich, Switzerland
[3] Royal Holloway, University of London, Egham, United Kingdom

Abstract. Most security models for authenticated key exchange (AKE) do not explicitly model the associated *certification system*, which includes the certification authority (CA) and its behaviour. However, there are several well-known and realistic attacks on AKE protocols which exploit various forms of malicious key registration and which therefore lie outside the scope of these models. We provide the first systematic analysis of *AKE security incorporating certification systems* (ASICS). We define a family of security models that, in addition to allowing different sets of standard AKE adversary queries, also permit the adversary to register arbitrary bitstrings as keys. For this model family we prove generic results that enable the design and verification of protocols that achieve security even if some keys have been produced maliciously. Our approach is applicable to a wide range of models and protocols; as a concrete illustration of its power, we apply it to the CMQV protocol in the natural strengthening of the eCK model to the ASICS setting.

Keywords: authenticated key exchange (AKE), unknown key share (UKS) attacks, certification authority (CA), invalid public keys, PKI

1 Introduction

After public key encryption and digital signatures, authenticated key establishment (AKE) is perhaps the most important public key primitive. From a real-world perspective, AKE protocols relying on public key techniques are widely deployed in systems that are used every day by billions of users, including systems such as TLS, IPsec, SSH, and various single sign-on systems. From a theoretical perspective, formal, cryptographically sound modelling for AKE protocols began in the symmetric setting with the seminal work of Bellare and Rogaway [4], and was later extended to the public key setting [6]. Since then, there has been a large body of work in this tradition, and many additions and modifications have been proposed. The most prominent current models in this tradition [3, 12, 25, 33] strengthen or add to the required security properties, cover different protocol classes, and strengthen adversary powers.

Despite intensive study over two decades, important elements of AKE protocols have not been sufficiently modelled, preventing our deeper understanding

of this important primitive and limiting its applicability to real-world protocols. Specifically, the public key infrastructure (PKI) needed to support the authenticity of public keys in AKE, and the interactions between the certification authority (CA), honest parties, and the adversary, are rarely modelled. Rather, with exceptions as noted below, in typical AKE models and proofs it is assumed that all public keys are honestly generated and authentically distributed at the start of the security game, and that there is a single key per party; certificates are excluded from the model. The adversary can corrupt parties, learning all their secrets, but has limited ability to register malicious keys. Roughly speaking, this modelling approach corresponds to an ideal CA, who zealously generates perfect key pairs and securely distributes them to the correct parties.

However, CAs in the real world simply do not operate in such rigorous ways. They have differing strengths of procedures for checking claimed identities[4], so malicious parties might in some cases get arbitrary public keys certified against identifiers of their choice. The most egregious examples involve CAs who, either willingly, under coercion, or as a result of security compromises, have issued certificates for keys and identifiers that they should not have.[5] CAs following best-practices may require that a user requesting a certificate submit a certificate signing request to the CA. This involves the user self-signing the data that is to be certified. Various standards [1, 2, 32] include other approaches to providing *proofs of possession*. However, even these basic tests of private key ownership are not mentioned in industry guidelines issued by the CA/Browser Forum [10, 11]. Furthermore, these procedures all fall short of the proofs of knowledge [31] required to match what is assumed in typical AKE models. Thus, an attacker may be able to register another party's public key under his own identifier, or register a malformed key which then interacts with properly generated keys in an unfortunate way.

Critically, there are realistic attacks on AKE protocols which cannot be captured by AKE security models that omit CA and PKI aspects:

- Kaliski's *unknown key share* (UKS) attack [22] on early versions of MQV exploits the ability of the adversary to dynamically register a public key (which is valid and for which the adversary *does* know the secret key).
- The UKS attack on KEA described by Lauter and Mityagin [26, p. 380] exploits the adversary's ability to re-register some party's static public key as his own public key.
- Blake-Wilson and Menezes [8] introduced the *duplicate-signature key selection* (DSKS) attack on signature schemes: after observing a user's signature σ

[4] For example, issuance of Extended Validation (EV) certificates requires stronger identity-checking requirements than non-EV certificates, see `https://www.cabforum.org/certificates.html` for more details.

[5] In June and July 2011, Dutch CA DigiNotar was hacked [18], with the intruder taking control of all 8 of the CA's signing servers; at least 531 rogue certificates were then issued. In August 2011, TURKTRUST CA [17] issued special certificates with wildcard signing capabilities, allowing impersonation of any domain in the Internet. This was discovered, by coincidence, only 18 months later.

on a message m, the adversary E is able to compute a signature key pair $(\mathsf{sk}_E, \mathsf{vk}_E)$ (or sometimes just a verification key vk_E) such that σ is also E's signature on the message m. Now, for example, if the Station-to-Station (STS) protocol is implemented using a signature scheme that is vulnerable to DSKS attacks, and the adversary can register arbitrary public keys with the CA, then the protocol is vulnerable to an online UKS attack [8].

– In Lim and Lee small subgroup attacks [27], the adversary extracts information about a party's long-term secret key. Some of these attacks require registering invalid public keys with the CA before engaging in protocol runs with honest participants. Of particular note are the Lim–Lee-style attacks of Menezes and Ustaoglu [29] on the HMQV protocol [23].

We claim that to date there has been no *systematic* treatment in the literature of the behaviour of CAs with respect to public keys and identifiers chosen by the adversary. Our paper sets out to rectify this situation, providing a comprehensive and self-contained treatment of these features, as well as establishing generic results to make protocols resilient against such attacks.

Contributions. Our paper has three main contributions.

First, we present in Section 2 a framework for reasoning about the security of AKE protocols with respect to various CA key registration procedures. This framework allows us to capture several attacks based on adversarial key registration, including UKS attacks, small-subgroup attacks, attacks that occur when the CA does not check if public keys are registered twice, and attacks that occur when multiple public keys can be registered per identifier.

Second, we provide in Section 3 a generic approach to achieve strong security guarantees against adversaries that can register arbitrary public keys for certain types of protocols. In particular, we show how to transform Diffie–Hellman type AKE protocols that are secure in a model where only honest key registration is allowed into protocols that are secure even when adversaries can register arbitrary valid or invalid public keys. In such cases, security is still guaranteed for all sessions (that were considered *clean* or *fresh* in the base model) except those in which the peer's public key is valid but registered by the adversary.

Third, we demonstrate in Section 4 how our methodology can be used to establish strong security guarantees, even when the adversary can register arbitrary public keys, for concrete protocols such as CMQV, NAXOS, and `UP`, using CMQV as a running example. We provide in Section 5 recommendations for the design of protocols that are secure in our models.

Related work. The original computational model for key exchange of Bellare and Rogaway [4] has a long-lived key generator, which is used to initialise all parties' keys at the start of the game. This is a standard part of most computational models today. However, in common with several later models [12, 21, 24], the adversary cannot influence long-term keys: only honestly generated keys are considered. Starting with the 1995 model of Bellare and Rogaway [5] it was recognised that the adversary may be able to choose long-term keys for certain

parties, whether public keys or symmetric keys. It is possible to identify three different methods that have been used to model such an adversary capability.

1. The adversary can replace long-term keys by providing them as an input to a corrupt query. This was the method used originally by Bellare and Rogaway [5] and was subsequently used in the public key setting by others [7, 30].
2. The adversary is allowed to generate arbitrary keys for corrupted parties at any time during the protocol run [23].
3. An additional query is added specifically to set up a user with a new key chosen by the adversary [14,20,35]. This query is typically called establishparty and takes as input the user name and its long-term public key.

These methods allow the models to capture the Kaliski attack [22], which requires the adversary to register a new public key after certain protocol messages have been obtained. However, none of these currently used methods has the generality of our model and, in particular, all of them omit the following realistic features:

– registration of multiple public keys per user;
– flexible checking by certification authorities via a verification procedure;
– adversarial choice of public keys per session.

Special mention should also be made of the model of Shoup [33]. Unlike most popular AKE models today, it uses a simulatability definition of security comparing ideal and real world views. Security is defined to mean that for any real world adversary there is an ideal world adversary (benign by definition) such that the transcripts of the two are computationally indistinguishable. Real-world adversaries have the ability to assign users to public key certificates. Shoup's model has not been widely used and the examples in [33] are not fully worked through. Furthermore, the model cannot represent an adversary who obtains only ephemeral secret keys without knowing the long-term key of the same user and therefore cannot capture security properties common in more modern models.

Other works [13, 19] have considered the security of non-interactive key exchange (NIKE) in settings where the adversary can register arbitrary public keys, analogously to our ASICS setting for interactive key exchange. It is an interesting open problem to examine how the security models and constructions for NIKE [13, 19] can be built upon to achieve security in the ASICS setting.

Critically, all of the approaches mentioned above have only been used to establish results for a handful of specific protocols. In contrast, we establish *generic results* that facilitate the design and verification of AKE protocols, and that can be applied to a large class of protocols.

2 ASICS model family

In this section we define a parameterized AKE security model that allows for explicit modelling of the certification of public keys. Prominent AKE security frameworks can be instantiated in this family of models, as well as extensions that allow dynamic adversarial registration of arbitrary bitstrings as public keys.

Generally speaking, from a user's point of view, participation in key exchange encompasses three consecutive phases: First, users set up their individual key pairs;

more precisely, each user invokes a randomized algorithm KeyGen that outputs a fresh secret-key/public-key pair $(\mathsf{sk}, \mathsf{pk})$. Second, users contact a certification authority (CA) to get their keys certified: each user provides the CA with its identifier $\hat{P}$ and its public key pk, and obtains a certificate C that binds the identifier to the key. After completing these setup steps, in the third phase, users can engage in interactive sessions with other users to establish shared keys. To do so, they usually require knowledge of their own key pair $(\mathsf{sk}, \mathsf{pk})$, their identifier $\hat{P}$, and the corresponding certificate C. In addition to that, protocols may require a priori knowledge of (a subset of) the peer's public key pk', peer's identifier $\hat{Q}$, and peer's certificate C'. As we will see, our execution model is general enough to cover all these settings. To ease notation, we assume that public key pk and identifier $\hat{P}$ can be readily derived from any certificate C; we use notation $C.\mathsf{pk} = \mathsf{pk}$ and $C.\mathsf{id} = \hat{P}$ correspondingly.

Our work enables the modeling of different degrees of rigour in the checks of consistency and ownership of public keys pk presented to the CA. On the one hand, CAs could be pedantic with such verifications (e.g., require a proof of knowledge of the secret key corresponding to pk); on the other hand, CAs could also just accept any given bitstring pk as valid and issue a certificate on it. The ability to precisely assess the security of key establishment in the face of different CA behaviours is a key contribution of our new model family.

Definition 1. *An* ASICS protocol Π *consists of a set of domain parameters, a key generation algorithm* KeyGen*, a public key verification procedure* VP*, and the protocol description* π *that describes how key exchange protocol messages are generated and responded to as well as how the session key is derived.*

We denote by VP the specific *verification procedure* on public keys and identifiers that a considered CA deploys. As different checks on pk and $\hat{P}$ might require different levels of interaction between the registering user and the CA, we model it as a procedure, as opposed to a function. We require that VP is efficient and has binary output. Furthermore, we require that the CA issues the requested certificate only if VP outputs value 1; all certification requests where VP outputs value 0 are rejected. Note that, for simplicity, we only consider non-interactive verification procedures (i.e., two-message registration protocols) between the user and the CA. A more general treatment covering interactive verification procedures as well would introduce additional complexities to our framework.

Specific key exchange protocols might be insecure for one (liberal) instantiation of VP, and be secure for another (stricter) one. Note that CAs that do not perform any check on pk and $\hat{P}$ are modelled by a verification procedure VP that always outputs 1. A verification procedure that performs few checks may output 1 for at least all $\mathsf{pk} \in \mathbf{PK}$, where $\mathbf{PK}$ denotes the set of possible public keys output by KeyGen. Precisely, if the inputs of algorithm KeyGen are security parameter 1^k and randomness $r \in_R \{0,1\}^k$, then we define

$$\mathbf{PK} = \left\{ \mathsf{pk} \mid \text{there exists } r \in \{0,1\}^k \text{ such that } \mathsf{KeyGen}(1^k; r) = (\cdot, \mathsf{pk}) \right\} \ .$$

A verification procedure with high assurance may require a zero-knowledge argument that the requester knows the secret key corresponding to the public

key, and even that the key was generated verifiably at random. Note that we allow VP to keep an internal state between invocations; our model hence covers possible implementations of CAs that reject certification requests with public keys that have already been registered (e.g., for a different identifier).

2.1 Security model

At a high level, our model stipulates users that generate one or more keys, obtain certificates for these keys from a CA, and use keys and certificates to run (potentially concurrent) sessions of the key agreement protocol. Similar to other security models, the adversary controls all communication in these sessions, corrupts users at will to obtain their secret keys, and arbitrarily reveals established session keys. Innovative is the adversary's additional ability to steer the registration process with the CA: it can obtain from the CA valid certificates for public keys and identifiers of its choosing (as long as VP evaluates to 1), and provides users with such certificates.

To keep our model simple and comprehensible, we abstract away any forgeability issues of certificates and assume the following ideal functionality: no certificate will be considered valid unless it has been issued by the CA. We model this by letting the challenger keep a list $\mathcal{C}$ of all CA-issued certificates and by equipping users with a *certificate verification oracle* $\mathcal{O}_{CV}$ that checks membership in that list; concretely, we assume that $\mathcal{O}_{CV}(C) = 1 \Leftrightarrow C \in \mathcal{C}$. Of course, in concrete implementations, this oracle is replaced by an explicit local verification routine; for instance, if certification is implemented via a signature scheme, this will include its verification procedure.

Sessions and session state. Users, once they have created their keys and obtained corresponding certificates, can execute protocol sessions. Within a user, each such session is uniquely identified by a pair $s = (C, i)$, where C denotes the certificate used by the user (by himself) in that session, and i is a counter. The user maintains session-specific variables as indicated in Table 1. Some session variables are fixed upon session creation, whereas others can be assigned or updated during protocol execution. Some, such as pcert, status, and key, are considered to be outputs of the key establishment and might be used in higher-level protocols or applications. A session s has *accepted* if $s_{\text{status}} = \texttt{accepted}$.

Adversarial queries. The adversary interacts with users by issuing queries. The adversary can direct users to establish long-term key pairs and certificates ($\mathsf{kgen}, \mathsf{hregister}$), to initiate protocol sessions (create), and to respond to protocol messages (send). The adversary may be able to learn long-term keys ($\mathsf{corrupt}$), session-specific randomness ($\mathsf{randomness}$), or session keys ($\mathsf{session\text{-}key}$) from users. The adversary can also maliciously obtain certificates from the CA ($\mathsf{pkregister}, \mathsf{npkregister}$).

The queries in set $\mathcal{Q}_N = \{\mathsf{kgen}, \mathsf{hregister}, \mathsf{create}, \mathsf{send}\}$, defined as follows, model normal operation of the protocol; they are required in any security model. Initially, the auxiliary variables $\mathcal{HK}$, $\mathcal{C}$, $\mathcal{C}_h$, $\mathcal{C}_{\mathsf{pk}}$, and $\mathcal{C}_{\mathsf{npk}}$ are set to $\emptyset$.

acert	certificate of the actor (the user running this session)
pcert	certificate of this session's peer
role	taken role; either $\mathcal{I}$ (initiator) or $\mathcal{R}$ (responder)
sent	concatenation of all messages sent in this session
rcvd	concatenation of all messages received in this session
status	session status; either `active`, `accepted`, or `rejected`
key	key in $\{0,1\}^k$ established in this session
rand	randomness used in this session
data	any additional protocol-specific data

Table 1. Elements of session state

$\mathcal{Q}_N = \{\mathsf{kgen}, \mathsf{hregister}, \mathsf{create}, \mathsf{send}\}$	(Normal protocol behaviour)
$\mathcal{Q}_S = \{\mathsf{corrupt}, \mathsf{randomness}, \mathsf{session\text{-}key}\}$	(corruption of Secrets)
$\mathcal{Q}_R = \{\mathsf{pkregister}, \mathsf{npkregister}\}$	(adversarial key Registration)

Table 2. Overview of query sets. Additionally, there is a test-session query.

- kgen () By running algorithm KeyGen, a fresh key pair (sk, pk) is generated. Public key pk is returned to the adversary; secret key sk is stored for processing potential later queries corresponding to pk. The public key is added to the set of honestly generated keys: $\mathcal{HK} \leftarrow \mathcal{HK} \cup \{\mathsf{pk}\}$.
- hregister(pk, $\hat{P}$) The query requires that $\mathsf{pk} \in \mathcal{HK}$ and that VP outputs 1 on input pk[6] and $\hat{P}$; otherwise, it returns $\perp$. The public key pk is registered at the CA for the identifier $\hat{P}$. The resulting certificate C is added to the global set of certificates and to the set of honestly generated certificates: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$ and $\mathcal{C}_h \leftarrow \mathcal{C}_h \cup \{C\}$. The query returns C.
- create $(s = (C, i), r, [C'])$ The query requires that $C \in \mathcal{C}_h$, that a session with counter i for certificate C does not already exist, and that $r \in \{\mathcal{I}, \mathcal{R}\}$; otherwise, it returns $\perp$. A new session s is created for the user with public key $C.\mathsf{pk}$ and identifier $C.\mathsf{id}$. Session variables are initialized as

$$(s_{\text{acert}}, s_{\text{pcert}}, s_{\text{role}}, s_{\text{sent}}, s_{\text{rcvd}}, s_{\text{status}}, s_{\text{key}}) \leftarrow (C, \perp, r, \epsilon, \epsilon, \texttt{active}, \perp) \ .$$

  If the optional certificate C' is provided, we set $s_{\text{pcert}} \leftarrow C'$. In addition, a string in $\{0,1\}^k$ is sampled uniformly at random and assigned to s_{rand}; we assume that all randomness required during the execution of session s is deterministically derived from s_{rand}. The user also runs the initialization procedure for the key exchange protocol, which may further initialize its own (internal) state variable s_{data} and optionally generate a message M. If M was generated, set $s_{\text{sent}} \leftarrow M$, and return M. Otherwise, return $\perp$.
- send (s, M) The query requires that session s exists and that $s_{\text{status}} =$ `active`; otherwise, it returns $\perp$. The user continues the protocol execution for this session with incoming message M, which may optionally generate a response message M'. Next, s_{rcvd} is set to $(s_{\text{rcvd}} \parallel M)$ and, if M' is output, s_{sent} is set to $(s_{\text{sent}} \parallel M')$. The protocol execution may (re-)assign values to

[6] Reasonable implementations of VP output 1 on all keys $\mathsf{pk} \in \mathcal{HK}$, because $\mathcal{HK} \subseteq \mathbf{PK}$.

s_{status} and s_{key}, and to the session's internal state variable s_{data}. Also, if the value s_{pcert} was not provided to the create query, then protocol execution may assign a value to s_{pcert}. If M' was generated, return M'; otherwise return $\perp$.

The queries in set $\mathcal{Q}_S = \{\mathsf{corrupt}, \mathsf{randomness}, \mathsf{session\text{-}key}\}$ model the corruption of a user's secrets. Similar queries are found in other standard AKE models [4,12].

- $\mathsf{corrupt}(\mathsf{pk})$ The query requires $\mathsf{pk} \in \mathcal{HK}$; otherwise, it returns $\perp$. This query returns the secret key sk corresponding to public key pk.
- $\mathsf{randomness}(s)$ The query requires that session s exist; otherwise, it returns $\perp$. The query returns the randomness s_{rand}. This is similar to the ephemeral key reveal query in the eCK model [25].
- $\mathsf{session\text{-}key}(s)$ The query requires that session s exist and that $s_{\text{status}} =$ `accepted`; otherwise, it returns $\perp$. The query returns the session key s_{key}.

The hregister query introduced above only allows registration of keys $\mathsf{pk} \in \mathcal{HK}$, i.e., keys held by honest users. In contrast, the adversary can obtain certificates on *arbitrary* (valid) public keys using the following pkregister query. Going even further, the npkregister query allows registration of objects that are not even public keys (always assuming that VP outputs 1 on the candidate object). These queries will allow modelling Kaliski's attack on MQV [22] and small subgroup attacks [27], amongst others. We emphasize that the queries in set $\mathcal{Q}_R = \{\mathsf{pkregister}, \mathsf{npkregister}\}$ have no counterparts in standard definitions of key exchange security.

- $\mathsf{pkregister}(\mathsf{pk}, \hat{P})$ The query requires that $\mathsf{pk} \in \mathbf{PK}$ and that VP outputs 1 on input pk and $\hat{P}$; otherwise, it returns $\perp$. The public key pk is registered at the CA for identifier $\hat{P}$. The resulting certificate C is added to the global set of certificates and to the set of certificates generated through pkregister query: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$ and $\mathcal{C}_{\mathsf{pk}} \leftarrow \mathcal{C}_{\mathsf{pk}} \cup \{C\}$. The query returns C.
- $\mathsf{npkregister}(\mathsf{pk}, \hat{P})$ The query requires that $\mathsf{pk} \notin \mathbf{PK}$ and that VP outputs 1 on input pk and $\hat{P}$; otherwise, it returns $\perp$. The public key pk is registered at the CA for the identifier $\hat{P}$. The resulting certificate C is added to the global set of certificates and to the set of certificates generated through npkregister query: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$ and $\mathcal{C}_{\mathsf{npk}} \leftarrow \mathcal{C}_{\mathsf{npk}} \cup \{C\}$. The query returns C.

2.2 Security experiment

Using the above queries, we define a parameterized family of AKE security models. As is common in BR-style AKE models, we must restrict query usage so that the adversary cannot trivially win the security experiment. The conditions under which queries are disallowed are expressed by a *freshness* condition, which typically uses a *matching* condition to formalize intended partner sessions.

Definition 2 (Matching, freshness, ASICS model). *Let Π be an ASICS protocol. A* matching condition *M for Π is a binary relation on the set of sessions of Π. Let Q be a set of queries such that $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S \cup \mathcal{Q}_R$. A* freshness condition *F for (Π, Q) is a predicate (usually depending on a matching condition M) that takes a session of Π and a sequence of queries (including*

arguments and results) of a security experiment over queries in Q. We call $X = (M, Q, F)$ an ASICS model *for Π.*

Definition 3 gives two possible matching conditions. We will later give examples of freshness conditions, in Example 1 on the following page and in Section 4.

The intricacies of matching definitions in AKE protocols are explored in detail by Cremers [15]. Two issues are important here. First, there is a strong connection between the information used in a matching definition and the information used to compute the session key. Second, some protocols like the two-message versions of MQV and HMQV allow sessions to compute the same key even if they perform the same role, whereas other protocols such as NAXOS require the sessions that compute the same key to perform different roles. In the remainder of the paper we will use one of the definitions below, depending on the type of protocol.

Definition 3 (M1-matching, M2-matching). *Let s and s' denote two sessions of an ASICS protocol. We say that session s'* M1-matches *(or is* M1-matching*) session s if $s_{\text{status}} = s'_{\text{status}} =$* `accepted` *and*

$$\begin{aligned}&(s_{\text{acert}}.\mathsf{pk}, s_{\text{acert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id}, s_{\text{sent}}, s_{\text{rcvd}})\\ &= (s'_{\text{pcert}}.\mathsf{pk}, s'_{\text{pcert}}.\mathsf{id}, s'_{\text{acert}}.\mathsf{pk}, s'_{\text{acert}}.\mathsf{id}, s'_{\text{rcvd}}, s'_{\text{sent}})\end{aligned}$$

Similarly, we say that session s' M2-matches *(or is* M2-matching*) session s if s'* M1*-matches session s and $s_{\text{role}} \neq s'_{\text{role}}$.*

The goal of the adversary is to distinguish the session key of a fresh session from a completely random string. This is modelled through an additional query:

- test-session (s) This query requires that session s exists and that $s_{\text{status}} =$ `accepted`; otherwise, it returns $\perp$. A bit b is chosen at random. If $b = 1$, then s_{key} is returned. If $b = 0$, a random element of $\{0,1\}^k$ is returned.

Definition 4 (ASICS_X experiment). *Let Π be an ASICS protocol and $X = (M, Q, F)$ be an ASICS model. We define experiment ASICS_X, between an adversary E and a challenger who implements all users and the CA, as follows:*

1. *The experiment is initialized with domain parameters for security parameter k.*
2. *The adversary E can perform any sequence of queries from Q.*
3. *At some point in the experiment, E issues a* test-session *query for a session s that has accepted and satisfies F at the time the query is issued.*
4. *The adversary may continue with queries from Q, under the condition that the test session must continue to satisfy F.*
5. *Finally, E outputs a bit b' as E's guess for b.*

Definition 5 (ASICS_X advantage). *The adversary E wins the security experiment if it correctly outputs the bit b chosen in the* test-session *query. The* ASICS_X-advantage *of E is defined as* $\mathrm{Adv}^{\mathsf{ASICS}_X}_{\Pi,E}(k) = |2\Pr(b = b') - 1|$.

Definition 6 (ASICS security). *Let Π be an ASICS protocol and $X = (M, Q, F)$ be an ASICS model. Π is said to be* secure in ASICS model X *if, for all PPT adversaries E, it holds that*

1. *if two users successfully accept in M-matching sessions, then they both compute the same session key, and*
2. *E has no more than a negligible advantage in winning the ASICS_X experiment; that is, there exists a negligible function* negl *in the security parameter k such that* $\mathrm{Adv}^{\mathsf{ASICS}_X}_{\Pi,E}(k) \leq \mathrm{negl}(k)$.

Remark 1 (Implicit authentication). Note that the ASICS security definition, like eCK-style security definitions, only provides implicit peer authentication, meaning that the key could only be known by the peer, not explicit authentication that the peer actually was active in the session.

Example 1. Let us consider the following ASICS model as a concrete example. Let $X = (\mathsf{M1}, Q, F)$ be the ASICS model given by $Q = \mathcal{Q}_N \cup \{\mathsf{session\text{-}key}\} \cup \mathcal{Q}_R$ and F defined as follows. Given a sequence of queries and a session s, F holds if:

- no $\mathsf{session\text{-}key}(s)$ query has been issued, and
- for all sessions s' such that s' M1-matches s, no query $\mathsf{session\text{-}key}(s')$ has been issued, and
- no query $\mathsf{pkregister}(s_{\mathrm{pcert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id})$ has been issued.

The model X is an extension of a BR-like model with a CA that allows registration of arbitrary keys. If a protocol is secure in X, then it is secure even if the adversary can register arbitrary bitstrings as public keys, as long as the specific peer key used in the test session is not an adversary-generated valid public key.

2.3 Capturing attacks

We illustrate how several attacks exploiting the adversary's ability to register valid or invalid public keys can be captured in ASICS models.

Kaliski's online UKS attack against MQV [22]. Kaliski's attack against MQV can be captured in an ASICS model where the adversary can register a specific valid public key with his own identifier via a pkregister query. As the adversary knows the secret key corresponding to the registered public key, the attack cannot be prevented by VP requiring a proof-of-possession of the secret key.

UKS attack against KEA based on public-key re-registration [26, p. 380]. Suppose that public key pk has been honestly registered at the CA for some user with identifier $\hat{P}$ via the query $\mathsf{hregister}(\mathsf{pk}, \hat{P})$. In this UKS attack on the KEA protocol, the adversary re-registers the public key pk under his own identifier $\hat{L} \neq \hat{P}$ by issuing the query $\mathsf{pkregister}(\mathsf{pk}, \hat{L})$. The attack is prevented if VP checks for uniqueness of the public key and outputs 0 when the public key was certified before (as observed in [26, p. 381]). Note that the UKS attack can also be prevented by making the session key derivation depend on users' identifiers.

UKS attack against KEA+ based on impersonation attack. Lauter and Mityagin [26] produced the KEA+ protocol from the KEA protocol and Protocol 4 in [6] by incorporating the identifiers of the user and its peer in the session key computation to prevent UKS attacks; however, a similar but previously unreported UKS attack still works on the KEA+ protocol. This UKS attack

involves a type of impersonation attack [34, p. 3]: it requires the adversary to successfully impersonate a user to the CA who then issues a certificate containing the user's identifier, but the adversary's valid public key. We stress that the attack does not arise when only one public key per identifier can be registered. See the full version of this paper [9] for a more detailed description of the attack.

Online UKS attack on STS-MAC based on duplicate-signature key selection (DSKS) [8]. Suppose that the signature scheme employed in the STS-MAC protocol is vulnerable to DSKS attacks. The UKS attack on STS-MAC [8, p. 160] exploits the ability of the adversary to register a valid public key pk under his own identifier during the run of the protocol. More precisely, the adversary first intercepts a user's message containing a signature σ on message m. He then issues a query $\mathsf{pkregister}(\mathsf{pk}, \hat{L})$ such that σ is also a valid signature on m under pk. The query associates pk with the adversary's identifier $\hat{L}$. Since the adversary knows the secret key corresponding to pk, he obtains a certificate from the CA even if VP requires a proof-of-possession. Countermeasures to such UKS attacks via modification of the protocol are available [8].

Lim–Lee style attack against HMQV with DSA domain parameters, without validation of ephemeral public keys [28]. Let $G = \langle g \rangle$ denote a q-order subgroup of $\mathbb{Z}_p^*$, where q and p are prime and $(p-1)/q$ is smooth. The attack on two-pass HMQV [28, p. 5] can be captured in an ASICS model where the adversary is given access to the queries in the set $Q = \mathcal{Q}_N \cup (\mathcal{Q}_S \setminus \{\mathsf{corrupt}\}) \cup (\mathcal{Q}_R \setminus \{\mathsf{pkregister}\})$. In particular, the adversary can register invalid public keys via the $\mathsf{npkregister}$ query. This attack can be prevented by countermeasures such as requiring VP to include a group membership test on the public key submitted for certification, or by including group membership tests on both ephemeral and long-term public keys during protocol execution. Small-subgroup attacks may also exist in other settings, for instance in groups over elliptic curves.

3 Achieving ASICS security

We provide a modular approach to obtain provable ASICS security for certain types of protocols. We first show in Theorem 1 how a result from Kudla and Paterson [24, Theorem 2] can be adapted to incorporate adversarial registration of valid public keys. Then, in Theorem 2, we indicate how to transform protocols to achieve security in the presence of adversaries that can register arbitrary invalid public keys. We start by defining an adapted version of *strong partnering* [24].

Definition 7 (Strong partnering). *Let Π be an ASICS protocol, and let $X = (M, Q, F)$ be an ASICS model. We say that Π has* strong partnering in the ASICS_X experiment *if no PPT adversary, when attacking Π in the ASICS_X experiment, can establish two sessions s and s' of protocol Π holding the same session key without being M-matching, with more than negligible probability in the security parameter k.*

Given an ASICS model $X = (M, Q, F)$, we denote by cNR-X ("computational No-Reveals" for session keys, following [24]) the reduced *computational* ASICS_X

experiment which is similar to the ASICS_X experiment except that the adversary (a) is not allowed to issue session-key and test-session queries, (b) must pick a session that has accepted and satisfies F at the end of its execution, and (c) output the session key for this session. See Kudla and Paterson [24] for a more detailed description of reduced games.

Definition 8 (cNR-X security). *Let Π be an ASICS protocol and $X = (M, Q, F)$ be an ASICS model. Π is said to be* cNR-X-secure *if, for all PPT adversaries E, it holds that*

1. *if two users successfully accept in M-matching sessions, then they both compute the same session key, and*
2. *E has no more than a negligible advantage in winning the* cNR-X *experiment; that is, there exists a negligible function* negl *in the security parameter k such that $\mathrm{Adv}^{\mathsf{cNR}\text{-}X}_{\Pi,E}(k) \leq \mathrm{negl}(k)$, where $\mathrm{Adv}^{\mathsf{cNR}\text{-}X}_{\Pi,E}(k)$ is defined as the probability that E outputs (s, s_{key}) for a session s that has accepted and satisfies F.*

Our first theorem deals with the security of DH-type ASICS protocols, which are a generalization of DH-type AKE protocols of Cremers and Feltz [16] to include certificates and to explicitly identify session strings. This class of protocols includes the most prominent modern two-message AKE protocols.

Definition 9 (DH-type ASICS protocol). *A* DH-type ASICS protocol *is an ASICS protocol of the following form, specified by functions $f_{\mathcal{I}}, f_{\mathcal{R}}, F_{\mathcal{I}}, F_{\mathcal{R}}, H$:*

- *Domain parameters (G, g, q), where $G = \langle g \rangle$ is a group of prime order q generated by g.*
- $\mathsf{KeyGen}()$*: Choose $a \in_R [0, q-1]$. Set $A \leftarrow g^a$. Return secret key $\mathsf{sk} = a$ and public key $\mathsf{pk} = A$.*
- $\mathsf{VP}(x, \hat{P}) = 1$ *for all x and all $\hat{P}$ (i.e., the CAs do not perform any checks).*
- *The specification of how users respond to* create *and* send *queries as well as how the session key is computed, namely as the hash H of some string which we call the session string, is given in Figure 1.*

Theorem 1. *Let $X = (M, Q, F)$ be an ASICS model with $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S$. Let $Y = (M, Q', F')$ be the ASICS model where $Q' = Q \cup \{\mathsf{pkregister}\}$ and F' is defined as follows. A session s is said to satisfy F' if it satisfies F and no $\mathsf{pkregister}(s_{\mathrm{pcert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id})$ query has been issued. Let Π be a DH-type ASICS protocol. Suppose that*

- *Π has strong partnering in the ASICS_Y experiment,*
- cNR-X *security of the related protocol π (defined in the same way as Π except that the session key generated in π is the session string of Π (i.e., $s^{\pi}_{\mathrm{key}} = ss^{\Pi}$)) is probabilistic polynomial-time reducible to the hardness of the computational problem of some relation ϕ,*
- *the session string decisional problem in the ASICS_Y experiment for Π is polynomial-time reducible to the decisional problem of ϕ, and*
- *there is a polynomial-time algorithm that decides whether an arbitrary bitstring is an element of G,*

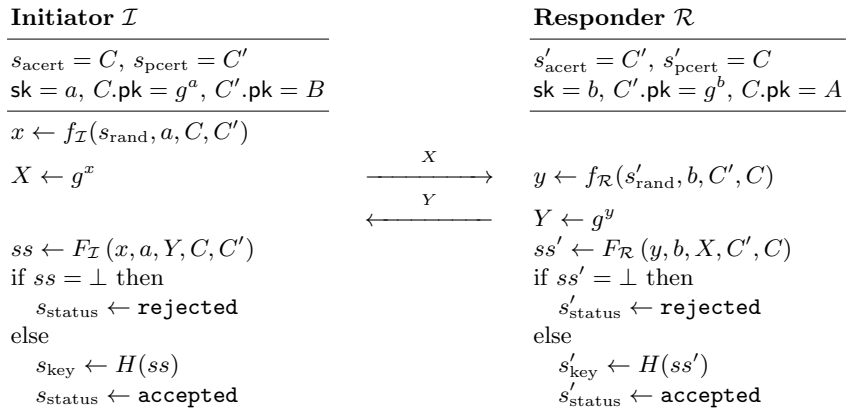

Fig. 1. Messages for generic DH-type ASICS protocol

then the security of Π in ASICS model Y is probabilistic polynomial-time reducible to the hardness of the gap problem of ϕ, if H is modelled as a random oracle.

In the cNR-X experiment of Theorem 1 the queries session-key and pkregister are not allowed, whereas in ASICS_Y both queries are allowed. Theorem 1 states that for any DH-type protocol Π, under certain conditions, it holds that security of the related protocol π in a reduced model (in which public keys can only be honestly registered) implies security of Π in the stronger non-reduced model that additionally captures adversarial registration of valid public keys.

The following theorem, which is applicable to a wider range of protocols than Theorem 1 (e.g., to three-message protocols such as UM [30] or HMQV-C [23]), allows us to achieve security against adversaries that can obtain certificates from the CA for invalid public keys by transforming the protocol to include a group membership test on the peer's public key. In contrast to Theorem 1, no additional requirement is imposed on the freshness condition of model Y.

Theorem 2. *Let $X = (M, Q, F)$ be an ASICS model with $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S \cup (\mathcal{Q}_R \setminus \{\mathsf{npkregister}\})$.*

Let Π be an ASICS protocol where the domain parameters (G, g, q), the key generation algorithm KeyGen *and the verification procedure* VP *are as in Definition 9.*

Let $f(\Pi)$ denote the ASICS protocol derived from Π by adding the following protocol step for each role of the protocol. Upon creation with (or, via send*, receipt of) the certificate C' to be used for the peer of session s, the user running session s checks whether the public key $C'.\mathsf{pk}$ belongs to the group G before continuing the execution of the protocol. In case the check fails, the protocol execution is aborted and s_{status} is set to* rejected.

Suppose that protocol Π is secure in ASICS model X and that there is a polynomial-time algorithm that decides whether an arbitrary bitstring is an element of G. Then the transformed protocol $f(\Pi)$ is secure in ASICS model $Y = (M, Q \cup \{\mathsf{npkregister}\}, F)$.

Combining both theorems, we obtain the following result.

Corollary 1. *Let Π be a DH-type ASICS protocol. Let $X = (M, Q, F)$ and $Y = (M, Q', F')$ be defined as in Theorem 1, and let the conditions of Theorem 1 hold with respect to protocol Π. Let $f(\Pi)$ denote the protocol derived from Π as specified in Theorem 2. Then the transformed protocol $f(\Pi)$ is secure in ASICS model $Z = (M, Q'', F')$, where $Q'' = Q' \cup \{\mathsf{npkregister}\}$, if H is modelled as a random oracle.*

Applying Corollary 1 to a concrete DH-type ASICS protocol that satisfies all the preconditions, we obtain a protocol that is secure in an ASICS model in which (a) sessions (including the test session) may use a certificate for the peer that resulted from an npkregister query, and (b) the certificate of the test session's peer was not the result of a pkregister query. The reader is referred to the full version of this paper [9] for detailed proofs of the above statements.

4 Applications

To illustrate the power of our generic approach, we examine in this section how to apply our technique to Ustaoglu's CMQV protocol [35]. CMQV is a modern DH-type protocol that is comparable in efficiency to HMQV, but enjoys a simpler security proof in the eCK model.

Our results allow us to analyse CMQV in a model that does not include session-key, pkregister, and npkregister queries, which simplifies the overall proof. We verify that CMQV meets the preconditions of Corollary 1, and conclude that a variant of CMQV with group membership test on the peer's public key is ASICS-secure in an eCK-like model. Similarly, our generic approach can be applied to other DH-type candidates such as NAXOS [25] and UP [36].

CMQV [35] was originally proven secure in the eCK model, where there is only one public key per identifier. In the ASICS setting, there is no such unique mapping between user identifiers and public keys. Hence, to be able to prove CMQV secure in the ASICS model, we need to include the public keys of the users in the session string to ensure that they have the same view of these keys when deriving the session key.

CMQV as a DH-type ASICS protocol. Two-pass CMQV can be stated as a DH-type ASICS protocol, by instantiating Definition 9 with the following functions. Let $\mathcal{H}_1 : \{0,1\}^k \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$, $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_q$, and $H : \{0,1\}^* \to \{0,1\}^k$ be hash functions. We define $f_{\mathcal{I}}, f_{\mathcal{R}}, F_{\mathcal{I}}, F_{\mathcal{R}}$ as:

$$f_{\mathcal{I}}(r, a, C, C') = \mathcal{H}_1(r, a)$$

$$F_{\mathcal{I}}(x, a, Y, C, C') = \begin{cases} \bot & \text{, if } Y \notin G \setminus \{1\} \\ ((YB^e)^{x+da} \parallel g^x \parallel Y \parallel C.\mathsf{id} \parallel A \parallel C'.\mathsf{id} \parallel B) & \text{, if } Y \in G \setminus \{1\} \end{cases}$$

$$f_{\mathcal{R}}(r, b, C', C) = \mathcal{H}_1(r, b)$$

$$F_{\mathcal{R}}(y, b, X, C', C) = \begin{cases} \bot & \text{, if } X \notin G \setminus \{1\} \\ ((XA^d)^{y+eb} \parallel X \parallel g^y \parallel C.\mathsf{id} \parallel A \parallel C'.\mathsf{id} \parallel B) & \text{, if } X \in G \setminus \{1\}, \end{cases}$$

where $d = \mathcal{H}_2(X \parallel C.\mathsf{id} \parallel C'.\mathsf{id})$, $e = \mathcal{H}_2(Y \parallel C.\mathsf{id} \parallel C'.\mathsf{id})$, $A = C.\mathsf{pk}$, $B = C'.\mathsf{pk}$; $\parallel$ denotes tagged concatenation to avoid ambiguity with variable-length strings.

We now show, using Corollary 1, that the resulting DH-type CMQV protocol is a secure ASICS protocol in an ASICS model with leakage queries corresponding to the eCK model.

ASICS model for eCK-like leakage. Define the ASICS model $\mathsf{eCK} = (\mathsf{M2}, Q, F)$ for eCK-like leakage [25] as follows. Let $Q = \mathcal{Q}_N \cup \mathcal{Q}_S$. Let F be the condition that a session s satisfies F if, for all sessions s' such that s' M2-matches s, none of the following conditions hold:

- a $\mathsf{session\text{-}key}(s)$ query has been issued;
- if s' exists:
  - a $\mathsf{session\text{-}key}(s')$ query has been issued;
  - both $\mathsf{corrupt}(s_{\mathrm{acert}}.\mathsf{pk})$ and $\mathsf{randomness}(s)$ queries have been issued;
  - both $\mathsf{corrupt}(s'_{\mathrm{acert}}.\mathsf{pk})$ and $\mathsf{randomness}(s')$ queries have been issued;
- if s' does not exist:
  - both $\mathsf{corrupt}(s_{\mathrm{acert}}.\mathsf{pk})$ and $\mathsf{randomness}(s)$ queries have been issued;
  - a $\mathsf{corrupt}(s_{\mathrm{pcert}}.\mathsf{pk})$ query has been issued.

Theorem 3. *Let f(CMQV) be the DH-type ASICS protocol derived from the* CMQV *protocol defined above, as specified in Theorem 2. If $\mathcal{H}_1, \mathcal{H}_2$ and H are modelled as random oracles, G is a group where the gap Diffie–Hellman assumption holds and membership in G is decidable in polynomial time, then f(CMQV) is secure in ASICS model $Z = (\mathsf{M2}, \mathcal{Q}_N \cup \mathcal{Q}_S \cup \mathcal{Q}_R, F')$, where a session s is said to satisfy F' if it satisfies the freshness condition F from the* eCK *model and no* $\mathsf{pkregister}(s_{\mathrm{pcert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id})$ *query has been issued.*

Proof (Sketch). We can readily show that CMQV satisfies the preconditions of Corollary 1 under the above formulation of the eCK model as an ASICS model:

1. *Strong partnering.* It is straightforward to see that CMQV has strong partnering in the $\mathsf{ASICS}_{\mathsf{eCK}'}$ game (where eCK' is derived from eCK as described in Theorem 1): since the session key in CMQV is computed via a random oracle, the probability that two sessions derive the same session key without using the same session string input to the random oracle is negligible.
2. $\mathsf{cNR\text{-}eCK}$*-security of the session string variant of CMQV.* This can be shown by an adaptation of Ustaoglu's original proof of CMQV. In large part, the main proof can be followed. However, a few simplifications can be made because the simulation need not answer session-key queries (so preventing key replication attacks and simulating sessions where the public key is a challenge value are easier).
3. *Hardness of the session string decision problem.* It can be easily seen that this is polynomial-time reducible to the decisional problem for Diffie–Hellman triples (U, V, W) by noting that the first component of the CMQV session string σ is equal to $g^{(y+eb)(x+da)} = g^{xy} g^{ady} g^{bex} g^{abde}$; the DDH values (U, V) can be injected into either (X, Y), (A, Y), (B, X), or (A, B), with W inserted into the corresponding part of σ, yielding a polynomial-time reduction.

Detailed proofs of each of the above claims can be found in the full version [9].

5 Lessons learned and recommendations

As we started our systematic investigation we assumed that certification authorities would need to perform some minimal checks on public keys to obtain secure KE protocols. Perhaps surprisingly, nearly all of the effort can be shifted to the protocols; and modern protocols often perform sufficient checks. In particular, our results provide formal foundations for some of the protocol-specific observations of Menezes and Ustaoglu [29]: checking that short- and long-term public keys are in the key space (i.e., in group G for DH-type protocols) is not superfluous.

Based on these observations, and given M public keys, N users may need to perform on the order of $M \times N$ such checks in total, even when caching the results. Reasoning purely about the overall amount of computation time used, one could consider moving the burden to the CAs. If the CAs only create certificates after a successful check, the CAs would only perform on the order of M checks in total. Depending on the deployment scenario, this might be a preferable alternative.

Similarly, CAs do not necessarily need to check uniqueness of public keys. As long as the key derivation involves the identifiers in an appropriate way, UKS attacks such as the one on KEA can be prevented. Even if public keys are associated with multiple identifiers, secrecy of the corresponding private key is sufficient to enable ASICS security for the honest user.

In general, our results further justify using as much information as possible in the key derivation function (KDF). This helps with establishing formal security proofs and it is also a prudent engineering principle. In particular, we recommend that in settings where users may have multiple long-term public keys, the input to the KDF should not only include the identifiers and the message transcript, but also the specific public keys used in the session.

We hope our work can serve as a foundation for the development of a range of protocols specifically designed to incorporate certification systems, offering different tradeoffs between efficiency and trust assumptions of the involved parties.

Acknowledgements. C.B. and D.S. are supported by Australian Research Council (ARC) Discovery Project DP130104304. C.C. and M.F. are supported by ETH Research Grant ETH-30 09-3. K.G.P. and B.P. are supported by a EPSRC Leadership Fellowship EP/H005455/1.

References

1. Adams, C., Farrell, S., Kause, T., Mononen, T.: Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210 (Proposed Standard) (Sep 2005), http://www.ietf.org/rfc/rfc4210.txt, updated by RFC 6712
2. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for key management — Part 1: General. NIST Special Publication (March 2007), http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf

3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer (2000)
4. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 232–249. Springer (1994)
5. Bellare, M., Rogaway, P.: Provably secure session key distribution: The three party case. In: 27th ACM STOC. pp. 57–66. ACM Press (1995)
6. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M. (ed.) 6th IMA International Conference on Cryptography and Coding. LNCS, vol. 1355, pp. 30–45. Springer (1997)
7. Blake-Wilson, S., Menezes, A.: Entity authentication and authenticated key transport protocols employing asymmetric techniques. In: Christianson, B., et al. (eds.) Security Protocols Workshop. Lecture Notes in Computer Science, vol. 1361, pp. 137–158 (1997)
8. Blake-Wilson, S., Menezes, A.: Unknown key-share attacks on the station-to-station (STS) protocol. In: Imai, H., Zheng, Y. (eds.) PKC'99. LNCS, vol. 1560, pp. 154–170. Springer (1999)
9. Boyd, C., Cremers, C., Feltz, M., Paterson, K.G., Poettering, B., Stebila, D.: ASICS: Authenticated key exchange security incorporating certification sytems. Cryptology ePrint Archive, Report 2013/398 (2013), http://eprint.iacr.org/
10. CA/Browser Forum: Baseline requirements for the issuance and management of publicly-trusted certificates, v1.1 (2011), https://cabforum.org/Baseline_Requirements_V1_1.pdf
11. CA/Browser Forum: Guidelines for the issuance and management of extended validation certificates, v1.4 (2012), https://cabforum.org/Guidelines_v1_4.pdf
12. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer (2001)
13. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer (2008)
14. Chatterjee, S., Menezes, A., Ustaoglu, B.: Combined security analysis of the one- and three-pass Unified Model key agreement protocols. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 49–68. Springer (2010)
15. Cremers, C.: Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In: Cheung, B.S.N., Hui, L.C.K., Sandhu, R.S., Wong, D.S. (eds.) ASIACCS 11. pp. 80–91. ACM Press (2011)
16. Cremers, C.J.F., Feltz, M.: Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 734–751. Springer (2012)
17. Ducklin, P.: The TURKTRUST SSL certificate fiasco — what really happened, and what happens next? (January 2013), http://nakedsecurity.sophos.com/2013/01/08/the-turktrust-ssl-certificate-fiasco-what-happened-and-what-happens-next/
18. FOX IT: Black Tulip: Report of the investigation into the DigiNotar Certificate Authority breach (2012), http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update/black-tulip-update.pdf
19. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Kurosawa, K., Hanaoka, G. (eds.) Public Key Cryptography. Lecture Notes in Computer Science, vol. 7778, pp. 254–271. Springer (2013)

20. Goldberg, I., Stebila, D., Ustaoglu, B.: Anonymity and one-way authentication in key exchange protocols. Designs, Codes and Cryptography 67(2), 245–269 (2013)
21. Jeong, I.R., Katz, J., Lee, D.H.: One-round protocols for two-party authenticated key exchange. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 04. LNCS, vol. 3089, pp. 220–232. Springer (2004)
22. Kaliski, B.S.: An unknown key-share attack on the MQV key agreement protocol. In: ACM Transactions on Information and System Security (TISSEC). vol. 4, pp. 275–288. ACM New York, NY, USA (August 2001)
23. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer (2005)
24. Kudla, C., Paterson, K.G.: Modular security proofs for key agreement protocols. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549–565. Springer (2005)
25. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer (2007)
26. Lauter, K., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer (2006)
27. Lim, C.H., Lee, P.J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 249–263. Springer (1997)
28. Menezes, A.: Another look at HMQV. Cryptology ePrint Archive, Report 2005/205 (2005), http://eprint.iacr.org/
29. Menezes, A., Ustaoglu, B.: On the importance of public-key validation in the MQV and HMQV key agreement protocols. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 133–147. Springer (2006)
30. Menezes, A., Ustaoglu, B.: Security arguments for the UM key agreement protocol in the NIST SP 800-56A standard. In: Abe, M., Gligor, V. (eds.) ASIACCS 08. pp. 261–270. ACM Press (2008)
31. Ristenpart, T., Yilek, S.: The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 228–245. Springer (2007)
32. Schaad, J.: Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF). RFC 4211 (Proposed Standard) (Sep 2005), http://www.ietf.org/rfc/rfc4211.txt
33. Shoup, V.: On formal methods for secure key exchange (version 4) (November 1999), revision of IBM Research Report RZ 3120 (April 1999) http://www.shoup.net/papers/skey.pdf
34. Turner, P., Polk, W., Barker, E.: ITL Bulletin for July 2012: Preparing for and responding to certification authority compromise and fraudulent certificate issuance (2012), http://csrc.nist.gov/publications/nistbul/july-2012_itl-bulletin.pdf (Accessed 12/03/2013)
35. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Designs, Codes and Cryptography 46(3), 329–342 (2008)
36. Ustaoglu, B.: Comparing SessionStateReveal and EphemeralKeyReveal for Diffie-Hellman protocols. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 183–197. Springer (2009)